目 录

Duang! 一座神奇的“桥”

◆ 作者：琦少

前言：

近年移动互联网好比 IT 界的一条浩瀚银河，安卓手机测试可谓其中最璀璨的一颗明珠，这里有一座你不得不熟悉的桥——ADB

曾见过不少相关的文章或培训，不乏空谈多，实战少；期望大，效果差。也见过很多测试同仁，预成大牛，却不得其法。今日我们秉承单点极致的精神。聚焦实用的技能，也思考下测试这条路，该如何走的深远。

安卓测试，乃如今移动测试领域中举足轻重的部分，相信也有不少同仁奋战在这片战场。工作中听闻最多的几个关键词“性能”，“自动化”，“稳定性”……掌握这些测试技能，或可成为高级测试工程师。也达到很多同仁眼中的成功，至少是成就感。那么这条路远嘛？难嘛？我说，不远，也不难，你和成功其实只隔一座桥——ADB。

ADB，或许很多人并不陌生，何谓 ADB，简言之：“adb 的全称为 Android Debug Bridge”就是起到调试桥的作用。最基本的安装，环境变量配置，相信大家也能查到诸多资料，而一些入门级的介绍更是犹如滔滔江水，连绵不绝。相信如下几个场景，或者命令大家都尝试过：

1) 安装一个安卓 APP: adb install
2) 卸载一个安卓 APP: adb uninstall
3) 查看设备连接: adb devices
4) 推送/拉去一个文件: adb push/pull
5) 重启手机: adb reboot

入门级命令十多个，相信大家看到这里很亲切。不错！这也是常考的一些笔试题。但是问题来了，又有多少人，到此就浅尝辄止了？！恐怕也是犹如黄河泛滥吧……

ADB 是一个非常实用的工具，尤其是对我们测试人员。什么才是进阶实践呢？您是否基于 ADB 做过“性能测试”，“自动化测试”，“稳定性测试”？姑且让我们从这三个场景看看这座桥（ADB）有多少神奇的潜力。

注：

1. 以下截图中代码，都来真实测试项目中的代码片段，必要地方我会加以解释。
2. h 是 Java 封装的一个类，主要作用是把 adb 命令发送到控制台执行。

很多高级语言都有相似方法，比如 php,或者 python

```
public class H {
    Process process = null;

    public ArrayList<String> ReadCmdLine(String cmd) {
        ArrayList<String> processList = new ArrayList<String>();
        try {
            process = Runtime.getRuntime().exec(
                    Configer.getAdbPath() + "adb " + cmd);
            BufferedReader input = new BufferedReader(new InputStreamReader(
                    process.getInputStream()));
            String line = "";
            while ((line = input.readLine()) != null) {
                processList.add(line);
            }
            input.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
        return processList;
    }
```

一、性能测试场景

安卓测试中，启动速度，安装包大小，内存占用，CPU 占用，耗电等是我们公认的重要性能指标。通过这座桥，这些都能实现。

例 1：测试 app 的启动速度：

【项目实战代码】:

```
// 启动app并获取启动时间
public int startAppForSpeed(String packageName, String mainActivity) {
    int startTime = 0;
    ArrayList<String> speedArray = h.ReadCmdLine("shell am start -W -n "
            + packageName + "/" + mainActivity);
    for (String line : speedArray) {
        if (line.split(":")[0].equals("TotalTime")) {
            System.out.println("启动时间->" + line.split(":")[1]);
            startTime = Integer.parseInt(line.split(":")[1].trim());
        }
    }
    return startTime;
}
```

【核心命令】:

```
adb shell am start -W -n com.cleanmaster.mguard/com.keniu.security.main.MainActivity
```

【效果分析】:

```
Starting: Intent { cmp=com.cleanmaster.mguard/com.keniu.security.main.MainActivi
ty }
Status: ok
Activity: com.cleanmaster.mguard/com.keniu.security.main.SplashingActivity
ThisTime: 701
TotalTime: 1170
Complete
```

其中的 ThisTime,和 TotalTime 就是我们关注的启动时间，单位是毫秒。

知道这个，结合你已经掌握的安装卸载命令，

再加上任何一门高级编程语言的基本语法就可以完成贵公司 app 启动速度专项测试。

提供思路如下（赶快动手试试吧）:

1) 设置安装包路径，如果有实力可以脚本完成定期去 FTP 服务器拉去最新可用安装包。
2) 安装
3) 启动并记录时间
4) 解析结果并入数据库。
5) 完成分析数据，横向对比，量化的把控启动速度的变化趋势。

例 2: 获取 app,安装后大小:

【项目实战代码】

```
// 获取安装后体积大小
public int getCmSize() {
    int result = 0;
    try {
        Runtime runtime = Runtime.getRuntime();
        Process process = runtime
                .exec(Configer.getAdbPath() + "adb shell ");
        DataOutputStream os = new DataOutputStream(
                process.getOutputStream());
        os.writeBytes("su" + "\n");
        os.flush();
        os.writeBytes("ls -l /data/app" + "\n");
        os.flush();
        BufferedReader input = new BufferedReader(new InputStreamReader(
                process.getInputStream()));
        String line = "";
        for (int i = 0; i < 100; i++) {
            line = input.readLine();
            if (line.contains("com.cleanmaster")) {
                String sizeString = line.subSequence(
                        line.lastIndexOf("system") + 7,
                        line.indexOf(":") - 14).toString();
                result = Integer.parseInt(sizeString.trim());
                break;
            }
        }
        input.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
    return result/1024;
}
```

【核心命令】:

```
adb shell su

ls -l /data/app
```

【效果分析】

```
root@ja3gchnduos:/ # ls -l /data/app
-rw-r--r-- system   system    2706806 2015-03-23 17:32 com.bst.ncr-1.apk
-rw-r--r-- system   system   13702828 2015-03-31 10:53 com.cleanmaster.mguard-1.
apk
-rw-r--r-- system   system    4099067 2015-03-24 14:38 com.qihoo.cleandroid_cn-1
.apk
-rw-r--r-- system   system    6385398 2015-03-23 17:32 com.sina.weibo-1.apk
-rw-r--r-- system   system   27777993 2015-03-23 17:32 com.tencent.mobileqq-1.ap
k
-rw-r--r-- system   system   10268328 2015-03-23 17:32 com.tencent.mtt-1.apk
-rw-r--r-- system   system    7301606 2015-03-25 17:34 com.wandoujia.phoenix2-1.
apk
-rw-r--r-- system   system     599895 2015-03-25 17:33 com.wandoujia.phoenix2.us
bproxy-1.apk
drwxrwxr-x system   system            2015-03-23 17:30 mcRegistry
```

其中 system 后面就是大小（单位是 B）顺便连安装时间也有了^_^。

租后面无疑是包名字，想针对具体的应用来看，自然也是可以的。

很简单，也很神奇。应用场景，你懂得。

例 3: 内存的获取:

【项目实战代码】

```
    // 获取系统剩余内存
    public int getSysFreeMem() {
        ArrayList<String> memArray = h.ReadCmdLine("shell cat /proc/meminfo");
        int freeMem = 0;
        for (int i = 1; i <= 3; i++) {
            freeMem += Integer.parseInt(memArray.get(i).split(":")[1].trim()
                    .split("kB")[0].trim());
        }
        System.out.println("可用内存->" + freeMem);
        return freeMem;
    }

// 获取系统总内存
public int getSysTotalMem() {
    ArrayList<String> memArray = h.ReadCmdLine("shell cat /proc/meminfo");
    System.out.println("系统总内存->"
            + Integer.parseInt(memArray.get(0).split(":")[1].trim().split(
                    "kB")[0].trim()));
    return Integer.parseInt(memArray.get(0).split(":")[1].trim()
            .split("kB")[0].trim());
}
```

```
    // 获取某一应用的内存占用
    public int getMemForApp(String packageName) {
        ArrayList<String> memArray = h
                .ReadCmdLine("shell dumpsys meminfo | grep " + packageName);
        int cost = 0;
        for (String line : memArray) {
            cost += Integer.parseInt(line.split("kB")[0].trim());
        }
        System.out.println("mem->" + cost / 2);
        return cost/2;

    }
```

【核心命令】:

```
adb shell cat /proc/meminfo
```

【效果分析】

```
127|root@ja3gchnduos:/ # cat /proc/meminfo
MemTotal:        1905504 kB
MemFree:          402376 kB
Buffers:           13972 kB
Cached:           554884 kB
SwapCached:            0 kB
Active:           708688 kB
Inactive:         417876 kB
Active(anon):     557760 kB
Inactive(anon):      836 kB
Active(file):     150928 kB
Inactive(file):   417040 kB
Unevictable:           0 kB
Mlocked:               0 kB
HighTotal:       1309688 kB
HighFree:          77996 kB
LowTotal:         595816 kB
LowFree:          324380 kB
SwapTotal:        511996 kB
SwapFree:         511996 kB
Dirty:                 4 kB
Writeback:             0 kB
AnonPages:        557804 kB
Mapped:           264488 kB
Shmem:               896 kB
Slab:              41300 kB
SReclaimable:      12424 kB
SUnreclaim:        28876 kB
```

远比你想想的要详细，惊呆了，有木有。其中前两项是我们要获取的总内存，和剩余内存，至于其他，按需获取。

【核心命令】:

```
# adb shell dumpsys meminfo |grep  com.cleanmaster
```

【效果分析】

```
1|root@ja3gchnduos:/ # dumpsys meminfo |grep  com.cleanmaster
    58753 kB: com.cleanmaster.mguard (pid 6304 / activities)
    26093 kB: com.cleanmaster.mguard:service (pid 5195)
              58753 kB: com.cleanmaster.mguard (pid 6304 / activities)
              26093 kB: com.cleanmaster.mguard:service (pid 5195)
```

不难看出吧 58753KB，就是猎豹清理大师当前的内存占用。

至于您想获取哪个应用，多久获取一次？合适获取。可以自己定义。总之内存这个常提到性能指标就通过一座“桥”实现了。

例 4: 电量测试

【项目实战代码】

```
// 获取系统电量信息
public HashMap<String, Integer> getSysBat() {
    ArrayList<String> batArray = h.ReadCmdLine("shell dumpsys battery");
    int level = 0;// 百分比
    int voltage = 0;// 电压
    int temp = 0;
    HashMap<String, Integer> batHah = new HashMap<String, Integer>();
    try {
        level = Integer.parseInt(batArray.get(7).split(":")[1].trim());
        voltage = Integer.parseInt(batArray.get(9).split(":")[1].trim());
        temp = Integer.parseInt(batArray.get(11).split(":")[1].trim());
    } catch (Exception e) {
        e.printStackTrace();
        System.out.println("看看手机插好了吗？");
    }
    System.out.println("电量->" + level);
    System.out.println("电压->" + voltage);
    System.out.println("电池温度->" + temp);
    batHah.put("level", level);
    batHah.put("voltage", voltage);
    batHah.put("temp", temp);
    return batHah;
}
```

【核心命令】:

```
adb shell dumpsys battery
```

【效果分析】

```
255|root@ja3gchnduos:/ # dumpsys battery
Current Battery Service state:
  AC powered: false
  USB powered: true
  Wireless powered: false
  status: 5
  health: 2
  present: true
  level: 100
  scale: 100
  voltage: 4325
  current now: 460
  temperature: 313
  technology: Li-ion _
```

不难看出，很容易解析到，我们要的电量，电压，温度。除此之外，你甚至会发

现，连充电方式，电池的健康程度，电池类型等都有。

想新到此，您已对 ADB 这座桥有了新的认知。

其实远远不仅如此。让我们继续领略一二。

二、功能自动化测试场景

一提到安卓安卓手机的自动化测试。恐怕就会想到，monkeyrunner，robotium,Athrun 等一系列的框架。其实根据项目的情况选择合适的测试技术也是一项很有学问的事情。在此不展开讨论。

说个具体场景。要检验猎豹清理大师的建议清理效果（该删除的文件被删除）

核心步骤:

1) 安装，启动，跳过协议页，规避弹窗:

adb install（安装）

adb shell am start（启动）

adb shell input keyevent 4(点击 back 键)

2) 进入垃圾清理，等待扫描完成，点击清理

adb shell input tap X Y(点击某点，横坐标 X，纵坐 Y 标)

3) 检查手机文件系统，完成效果比对。

adb shell ls

```
for (String line : new H().ReadCmdLine("shell ls " + filePath)) {
    if (line.contains("No such file or directory")) {
        flag = false;
        break;
    }
}
```

这无疑是一个典型的自动化测试场景。 核心的 ADB 命令如上，大家可以小试牛刀。

三、基本稳定性测试场景

提到稳定性，大家第一时间想到，monkey。不错，确实有一条命令:

adb shell monkey –n XX (包名) 1000（伪随机时事件数）

这无疑算是一种稳定性测试方法，但是常用的同学也不难发现它的缺点！

>随机性强，不容易复现。

>盲目性大，很容易点到别的应用。

>可控性差，运行很久未必重点测试预期的模块。

今天我们有这样一个测试场景，测试手机主要的模块，能正正常进入，并且没有崩溃。

核心点有三个:

1) 驱动手机进入指定模块;

ADB 的坐标点击，你懂得，就不赘述

2) 获取当前系统最上层的 ActivITy 名称。

【项目实战代码】

```
public String getNowActivity() {
    ArrayList<String> lines = getH().ReadCmdLine(
            "shell dumpsys activity | grep mFocusedActivity");
    String name = "";
    for (String l : lines) {
        name = l.split("u0")[1].trim().split("/")[1].trim().split(" ")[0]
                .trim();
    }
    return name;
}
```

【核心命令】:

```
adb shell dumpsys activity | grep mFocusedActivity
```

【效果分析】

```
1|root@ja3gchnduos:/ # dumpsys activity | grep mFocusedActivity
  mFocusedActivity: ActivityRecord{431497c0 u0 com.cleanmaster.mguard/com.keniu.security.main.MainActivity t4}
```

3) 结果比实际最上层 ActivITy 名称，和预期比较，判断稳定性，记录结果。

字符串比较，留给大家自己发挥。

至此，我们多少重新认识了下，似曾相识的 ADB。其实安卓测试，认真的走过这座

“桥”漠然回首的时候，你会发现，自己经找到晋身高级测试工程师的门路了。当然由于时间和字数的限制。今日只能带大家走马观花的领略下，这座神器的“桥”。工作中如果有疑难的测试场景，不妨想起 ADB，或许一些问题就迎刃而解，修行在个人，希望大家一起更上一层楼。

最后再次如何在测试这条路上走的更深，更远。虽说命题有些宽泛，但是有些方法论可以参考

>在某个领域钻的够深，你就是专家。

>简单的事情做出新意，你就是大牛。

>茫然的领域探出条道路，你就是领袖。

和大家共勉。

LoadRunner 脚本优化之
——参数化迭代介绍

◆ 作者：曹承臻

在 LoadRunner 的脚本优化时，有时发送给服务器的请求参数化时，服务器返回的内容也会和参数化的内容相对应，例如发送的请求带有查询 key=123，则服务器也会返回含有 123 相关的内容。这时我们在使用检查点检查服务器参数化返回的数据正确性时，通常也会用到和服务器同样的参数。

```
web_reg_find("Search=Body",
    "Text={NewParam}",
    LAST);

web_custom_request("address",
    "URL=http://aurora.ie.sogou.com/address?key={NewParam}",
    "Resource=0",
    "RecContentType=text/html",
    "Referer=",
    "Snapshot=t4.inf",
    "Mode=HTML",
    LAST);
```

这样在每次迭代过程中，每次都会取不同的值，完成检查过程。

```
address.c(8): Notify: Parameter Substitution: parameter "NewParam" = "1a"
address.c(8): Registering web_reg_find was successful [MsgId: MMSG-26390]
address.c(12): Notify: Parameter Substitution: parameter "NewParam" = "1a"
address.c(12): Registered web_reg_find successful for "Text=1a" (count=2) [MsgId: MMSG-26364]
address.c(12): web_custom_request("address") was successful, 138 body bytes, 155 header bytes [MsgId
Ending action address.
Starting action uninstallres.
Ending action uninstallres.
Ending iteration 1.
Starting iteration 2.
Notify: Next row for parameter NewParam = 2 [table = NewParam].
Notify: Getting new value for parameter 'NewParam': table = 'NewParam.dat' column = '0' row = '2'.
Starting action address.
address.c(8): Notify: Parameter Substitution: parameter "NewParam" = "2b"
address.c(8): Registering web_reg_find was successful [MsgId: MMSG-26390]
address.c(12): Notify: Parameter Substitution: parameter "NewParam" = "2b"
address.c(12): Registered web_reg_find successful for "Text=2b" (count=12) [MsgId: MMSG-26364]
address.c(12): web_custom_request("address") was successful, 376 body bytes, 155 header bytes [MsgId
Ending action address.
Starting action uninstallres.
Ending action uninstallres.
Ending iteration 2.
Starting iteration 3.
Notify: Next row for parameter NewParam = 3 [table = NewParam].
Notify: Getting new value for parameter 'NewParam': table = 'NewParam.dat' column = '0' row = '3'.
Starting action address.
address.c(8): Notify: Parameter Substitution: parameter "NewParam" = "3c"
address.c(8): Registering web_reg_find was successful [MsgId: MMSG-26390]
address.c(12): Notify: Parameter Substitution: parameter "NewParam" = "3c"
address.c(12): Registered web_reg_find successful for "Text=3c" (count=9) [MsgId: MMSG-26364]
```

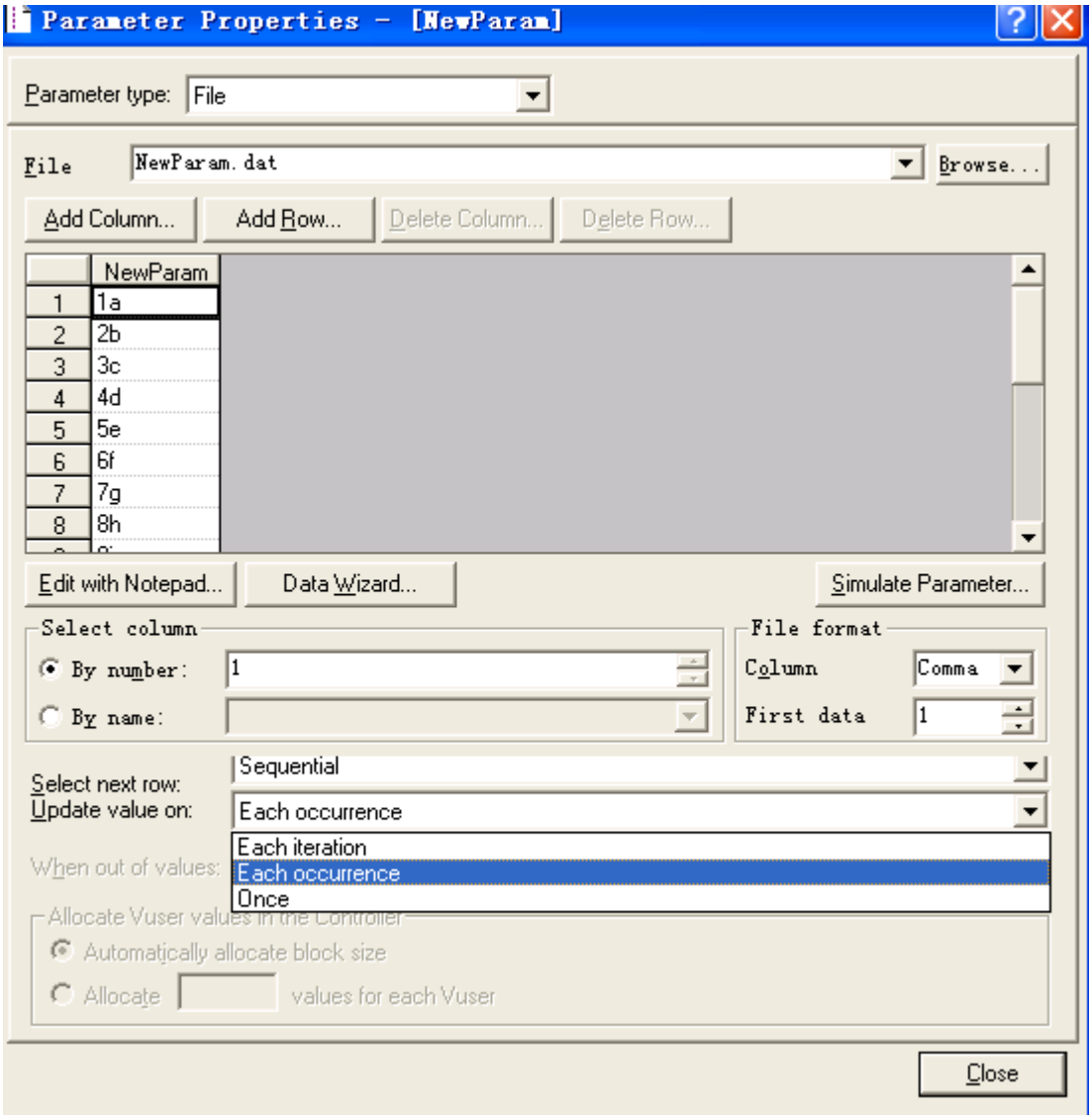
但是如果基于实际场景设计的脚本是：在一个迭代周期内，此 action 需要循环多次，于是引入了 block 块。将此 action 加入到一个 block 块中，设置循环次数为 2。再次运行一下，得到这样的结果：

```
address.c(8): Notify: Parameter Substitution: parameter "NewParam" = "1a"
address.c(8): Registering web_reg_find was successful [MsgId: MMSG-26390]
address.c(12): Notify: Parameter Substitution: parameter "NewParam" = "1a"
address.c(12): Registered web_reg_find successful for "Text=1a" (count=2) [MsgId: MMSG-26364]
address.c(12): web_custom_request("address") was successful, 138 body bytes, 155 header bytes [MsgId: MMSG-2638
Ending action address.
Starting action address.
address.c(8): Notify: Parameter Substitution: parameter "NewParam" = "1a"
address.c(8): Registering web_reg_find was successful [MsgId: MMSG-26390]
address.c(12): Notify: Parameter Substitution: parameter "NewParam" = "1a"
address.c(12): Registered web_reg_find successful for "Text=1a" (count=2) [MsgId: MMSG-26364]
address.c(12): web_custom_request("address") was successful, 138 body bytes, 155 header bytes [MsgId: MMSG-2638
Ending action address.
Starting action uninstallres.
Ending action uninstallres.
Ending iteration 1.
Starting iteration 2.
Notify: Next row for parameter NewParam = 2 [table = NewParam].
Notify: Getting new value for parameter 'NewParam': table = 'NewParam.dat' column = '0' row = '2'.
Starting action address.
address.c(8): Notify: Parameter Substitution: parameter "NewParam" = "2b"
address.c(8): Registering web_reg_find was successful [MsgId: MMSG-26390]
address.c(12): Notify: Parameter Substitution: parameter "NewParam" = "2b"
address.c(12): Registered web_reg_find successful for "Text=2b" (count=12) [MsgId: MMSG-26364]
address.c(12): web_custom_request("address") was successful, 376 body bytes, 155 header bytes [MsgId: MMSG-2638
Ending action address.
Starting action address.
address.c(8): Notify: Parameter Substitution: parameter "NewParam" = "2b"
address.c(8): Registering web_reg_find was successful [MsgId: MMSG-26390]
address.c(12): Notify: Parameter Substitution: parameter "NewParam" = "2b"
```

可以发现：在每个 ITerations 时，参数化的值都会更新，但是在单个 ITerations 的多次 block 循环时，每次取得的参数化的值是一样的，问题来了：如果让每次 block 块的循环也取得不一样的值呢？

查了下资料，发现通过参数化的设置可以做到。

打开参数化设置框



其中 Update value 提供了三个可选择方式:

Each ITeration: 每次迭代更新参数值。

Each occurrence: 每次出现此参数时，更新此参数值。

Once: 只取一次，一直这样用下去。

看到这里后，果断使用 Each occurrence 方式，坐等运行大吉啦~

```
Running Vuser...
Starting iteration 1.
Starting action address.
address.c(8): Notify: Getting new value for parameter 'NewParam': table = 'NewParam.dat' column = '0' row = '1'.
address.c(8): Notify: Parameter Substitution: parameter "NewParam" = "1a"
address.c(8): Registering web_reg_find was successful   [MsgId: MMSG-26390]
address.c(12): Notify: Next row for parameter NewParam = 2 [table  = NewParam].
address.c(12): Notify: Getting new value for parameter 'NewParam': table = 'NewParam.dat' column = '0' row = '2'.
address.c(12): Notify: Parameter Substitution: parameter "NewParam" = "2b"
address.c(12): Error -26366: "Text=1a" not found for web_reg_find   [MsgId: MERR-26366]
address.c(12): web_custom_request("address") highest severity level was "ERROR", 376 body bytes, 155 header bytes
Ending action address.
Starting action address.
address.c(8): Notify: Next row for parameter NewParam = 3 [table  = NewParam].
address.c(8): Notify: Getting new value for parameter 'NewParam': table = 'NewParam.dat' column = '0' row = '3'.
address.c(8): Notify: Parameter Substitution: parameter "NewParam" = "3c"
address.c(8): Registering web_reg_find was successful   [MsgId: MMSG-26390]
address.c(12): Notify: Next row for parameter NewParam = 4 [table  = NewParam].
address.c(12): Notify: Getting new value for parameter 'NewParam': table = 'NewParam.dat' column = '0' row = '4'.
address.c(12): Notify: Parameter Substitution: parameter "NewParam" = "4d"
address.c(12): Error -26366: "Text=3c" not found for web_reg_find   [MsgId: MERR-26366]
address.c(12): web_custom_request("address") highest severity level was "ERROR", 58 body bytes, 154 header bytes
Ending action address.
Starting action uninstallres.
Ending action uninstallres.
Ending iteration 1.
Starting iteration 2.
Starting action address.
address.c(8): Notify: Next row for parameter NewParam = 5 [table  = NewParam].
address.c(8): Notify: Getting new value for parameter 'NewParam': table = 'NewParam.dat' column = '0' row = '5'.
```

红红的提示告诉我，脚本没有跑过，仔细看了下，发现了问题原因:

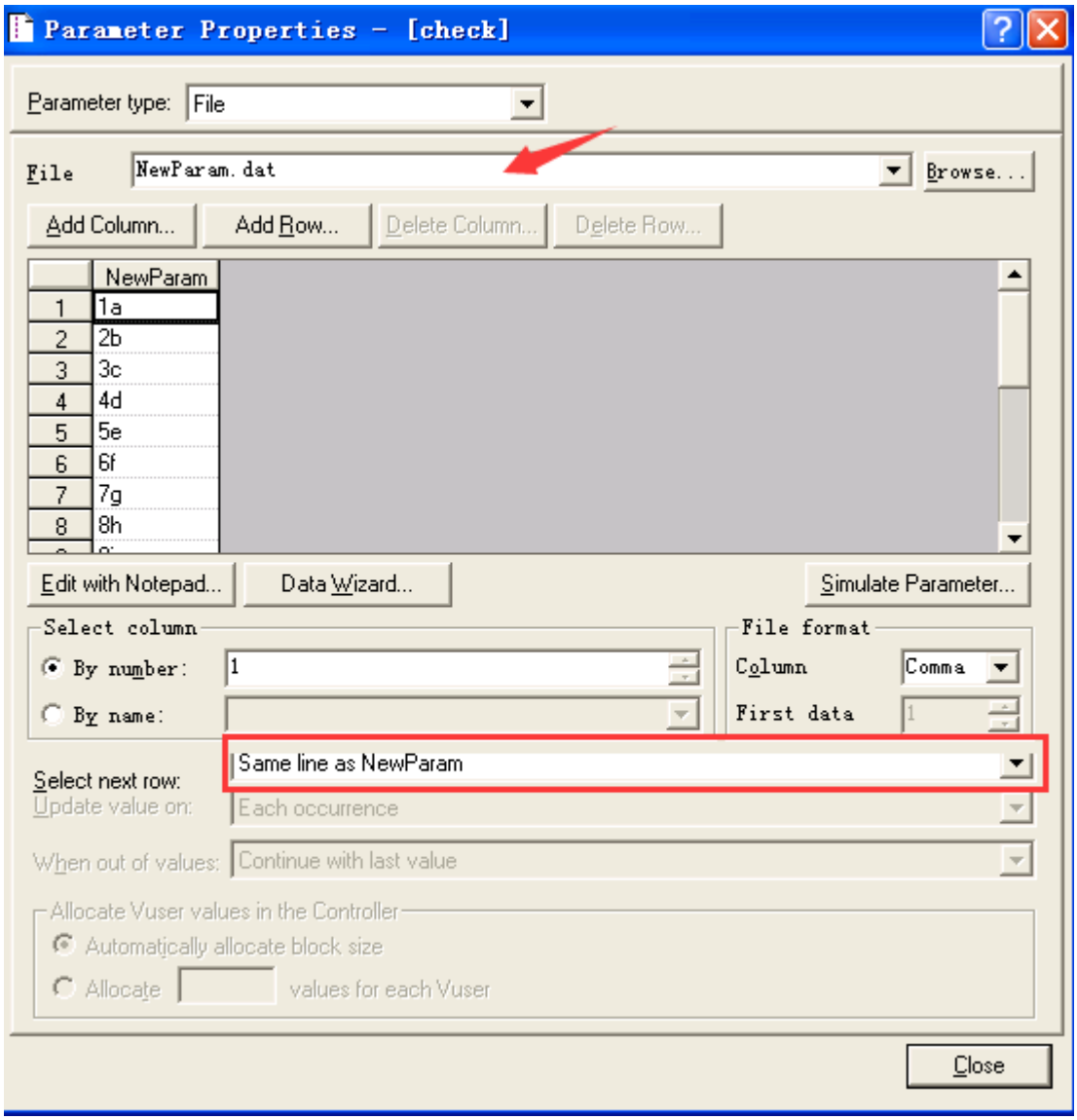
由于参数化的策略是每次出现都会重新取值，这样在 find 函数和真正调用的函数中，都会调用不同的值，于是就会出现检查的值和实际运行的值总是取的不同的情况。那这种情况应该怎么破呢？

又要用参数化，但检查的函数和运行的函数要用同一参数，但是不能用同一参数配置。可以用以下方法解决:

对检查函数重新定义一个参数化变量“check”，在其参数配置中如下设置:

选择需要运行的参数化数据表

Select next row 策略使用“Same line as NewParam”



重新运行了一下，发现还是会报错:

```
Run-Time Settings file: "G:\Program Files\Mercury\scripts\test script\\default.cfg"      [MsgId: MMSG-27141]
Ending action vuser_init.
Running Vuser...
Starting iteration 1.
Starting action address.
address.c(8): Notify: Next row for parameter check = 1 [table  = check].
address.c(8): Notify: Getting new value for parameter 'check': table = 'NewParam.dat' column = '0' row = '1'.
address.c(8): Notify: Parameter Substitution: parameter "check" =  "1a"
address.c(8): Registering web_reg_find was successful    [MsgId: MMSG-26390]
address.c(12): Notify: Getting new value for parameter 'NewParam': table = 'NewParam.dat' column = '0' row = '1'.
address.c(12): Notify: Parameter Substitution: parameter "NewParam" =  "1a"
address.c(12): Registered web_reg_find successful for "Text=1a" (count=2)    [MsgId: MMSG-26364]
address.c(12): web_custom_request("address") was successful, 138 body bytes, 155 header bytes    [MsgId: MMSG-2638
Ending action address.
Starting action address.
address.c(8): Notify: Next row for parameter check = 1 [table  = check].
address.c(8): Notify: Next row for parameter check = 1 [table  = check].
address.c(8): Notify: Getting new value for parameter 'check': table = 'NewParam.dat' column = '0' row = '1'.
address.c(8): Notify: Parameter Substitution: parameter "check" =  "1a"
address.c(8): Registering web_reg_find was successful    [MsgId: MMSG-26390]
address.c(12): Notify: Next row for parameter NewParam = 2 [table  = NewParam].
address.c(12): Notify: Getting new value for parameter 'NewParam': table = 'NewParam.dat' column = '0' row = '2'.
address.c(12): Notify: Parameter Substitution: parameter "NewParam" =  "2b"
address.c(12): Error -26366: "Text=1a" not found for web_reg_find    [MsgId: MERR-26366]
address.c(12): web_custom_request("address") highest severity level was "ERROR", 376 body bytes, 155 header bytes
Ending action address.
Starting action uninstallres.
Ending action uninstallres.
Ending iteration 1.
Starting iteration 2.
```

分析了下，由于 check 是 same line wITh “NewParam”，而 NewParam 是后运行的，这样就会导致了 check 总是会慢半拍。于是将这两个变量对调了一下：check 变量使用参数化并 Update value=Each occurrence，NewParam 变量 same line wITh “check”，重新运行了下，发现没有错误了。

```
address.c(8): Notify: Getting new value for parameter 'check': table = 'NewParam.dat' column = '0' row = '1'.
address.c(8): Notify: Parameter Substitution: parameter "check" =  "1a"
address.c(8): Registering web_reg_find was successful    [MsgId: MMSG-26390]
address.c(12): Notify: Next row for parameter NewParam = 1 [table  = NewParam].
address.c(12): Notify: Getting new value for parameter 'NewParam': table = 'NewParam.dat' column = '0' row = '1'.
address.c(12): Notify: Parameter Substitution: parameter "NewParam" =  "1a"
address.c(12): Registered web_reg_find successful for "Text=1a" (count=2)    [MsgId: MMSG-26364]
address.c(12): web_custom_request("address") was successful, 138 body bytes, 155 header bytes    [MsgId: MMSG-2638
Ending action address.
Starting action address.
address.c(8): Notify: Next row for parameter check = 2 [table  = check].
address.c(8): Notify: Getting new value for parameter 'check': table = 'NewParam.dat' column = '0' row = '2'.
address.c(8): Notify: Parameter Substitution: parameter "check" =  "2b"
address.c(8): Registering web_reg_find was successful    [MsgId: MMSG-26390]
address.c(12): Notify: Next row for parameter NewParam = 2 [table  = NewParam].
address.c(12): Notify: Getting new value for parameter 'NewParam': table = 'NewParam.dat' column = '0' row = '2'.
address.c(12): Notify: Next row for parameter NewParam = 2 [table  = NewParam].
address.c(12): Notify: Getting new value for parameter 'NewParam': table = 'NewParam.dat' column = '0' row = '2'.
address.c(12): Notify: Parameter Substitution: parameter "NewParam" =  "2b"
address.c(12): Registered web_reg_find successful for "Text=2b" (count=12)      [MsgId: MMSG-26364]
address.c(12): web_custom_request("address") was successful, 376 body bytes, 155 header bytes    [MsgId: MMSG-2638
Ending action address.
Starting action uninstallres.
Ending action uninstallres.
Ending iteration 1.
Starting iteration 2.
Starting action address.
address.c(8): Notify: Next row for parameter check = 3 [table  = check].
address.c(8): Notify: Getting new value for parameter 'check': table = 'NewParam.dat' column = '0' row = '3'.
address.c(8): Notify: Parameter Substitution: parameter "check" =  "3c"
address.c(8): Registering web_reg_find was successful    [MsgId: MMSG-26390]
address.c(12): Notify: Next row for parameter NewParam = 3 [table  = NewParam].
address.c(12): Notify: Getting new value for parameter 'NewParam': table = 'NewParam.dat' column = '0' row = '3'.
```

附：参数化不同方式详解：

http：//blog.chinaunix.net/uid-9236609-id-2005877.html

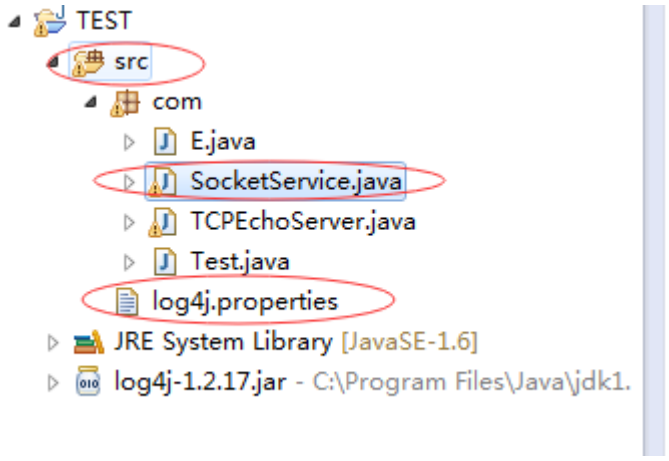
Socket 挡板之 Java 开发

◆作者：徐继伟

看云层的《性能测试进阶指南 II》,看到有一章节是关于挡板测试的，书中的挡板程序是用 python 写的，这对于我来说是一个新的语言，我就开始想，能不能用我熟悉的语言实现呢？我就尝试用 java 语言写了一个关于 Socket 挡板程序，该程序还用了 log4j 的日志函数。

简单说一下代码，首先开启一个端口，这里的端口可以自己定义，如果没有定义就默认 23011 端口，用 LR 建立一个 socket 连接，发送任意数据，挡板程序就会返回固定的报文。

目录树：



SocketService.java 源码如下：

```
package com;

import java.io.*;

import java.net.ServerSocket;

import java.net.Socket;

import java.net.SocketAddress;

import java.net.URL;

import org.apache.log4j.*;
```

```
public class SocketService {
  ServerSocket serverSocket;
  int thread=1;
  Logger logger = Logger.getLogger(SocketService.class);

  public SocketService(){
      try {
          serverSocket=new ServerSocket(23011,500);
          System.out.println("无参数，采用默认端口。端口 23011 已打开");
          while(true){
              Socket socket=serverSocket.accept();
              SocketAddress clientAdress = socket.getRemoteSocketAddress();//获取连接到服务器的 ip
              System.out.println("客户端：  "+clientAdress);
              logger.info("开启线程"+thread);
              SocketServiceThread sst=new SocketServiceThread(socket,clientAdress);
              sst.start();
              thread++;
          }
      } catch (IOException e) {
          e.printStackTrace();
      }finally
      {
          System.out.println("ddddddddddddd"+thread);
      }
  }
  public SocketService(int port){
      try {
          serverSocket=new ServerSocket(port,500);
          System.out.println("端口"+port+"已打开");
```

```
                while(true){
                    Socket socket=serverSocket.accept();
                    SocketAddress clientAdress = socket.getRemoteSocketAddress();//获取连接到服务器的 ip
                    System.out.println("客户端:  "+clientAdress);
                    logger.info("开启线程"+thread);
                    SocketServiceThread sst=new SocketServiceThread(socket,clientAdress);
                    sst.start();
                    thread++;
                }
            } catch (IOException e) {
                e.printStackTrace();
            }finally
            {
                System.out.println("ddddddddddd"+thread);
            }
        }

        class SocketServiceThread extends Thread{
            Socket socket;
            SocketAddress adress;
            InputStream in_tmp;
            OutputStream out_tmp;
            int recvMsgSize;
            int i=1;
            boolean done;
            String str1;
            String str=new String("<?xml version=\"1.0\" encoding=\"UTF-8\" ?><Banksh><Message
id=\"8772b9853272c7da\"><CSRes id=\"CPRes\"><version>1.0.1</version> <coreOrderNum />
<instId>123789</instId> <errorCode>S72_7040</errorCode> <errorMessage>主账号没找到</errorMessage>
<orderNum>20150127135149</orderNum> <amount>10</amount> <eacct>623185000896867070</eacct>
```

```
<serialNo>201501271000001548</serialNo> <date>20150127 13： 52： 15</date> <period /> <dlday />
<leftLoanPeriod /> <tranAmount>1000</tranAmount> <paymentMethod />
<prodParam>SuningCredITPay</prodParam> <subProdParam>InstallmentPay3</subProdParam>
<currency>156</currency> <stageRate /> <punishRate /> <recordNum /> <repayList />
</CSRes><Signature>Vz+y9Kf0ALo1m+s3i9oHMPaXbmxZx5U6biliR2g938WjsvcyABTfeaWxWMcsYTnl7Ss
mcwPK846XC/hBLyzclq/f9Bo03+TFC9RJe1Tc12+9mWtIsRPtul3wiZk68oARU76CVbnLjD8+jeqJHC3LvaWiJb
L4bMDoTFtUDTNVEhc=</Signature> </Message></Banksh>");//正确的响应报文
    // String str_err = new String("22222222222222222222<errorCode> 主 要 字 段 错 误 ！
</errorCode>4444444444444444444444444444444444444444444444444444");
            byte[] receiveBuf=new byte[1071];
            boolean runFlag=true;
            byte[] sendbuf_tmp=str.getBytes();
    // byte[] sendbuf_tmp_err=str_err.getBytes();
            int lenth =str.length();

    public SocketServiceThread(Socket socket,SocketAddress address){
            if(null==socket){
                runFlag=false;
                return;
            }
            this.socket=socket;
            this.adress=address;
        }

        public void run(){
            if(null==socket){
                System.out.println("socket is null");
                return;
            }
            try {
```

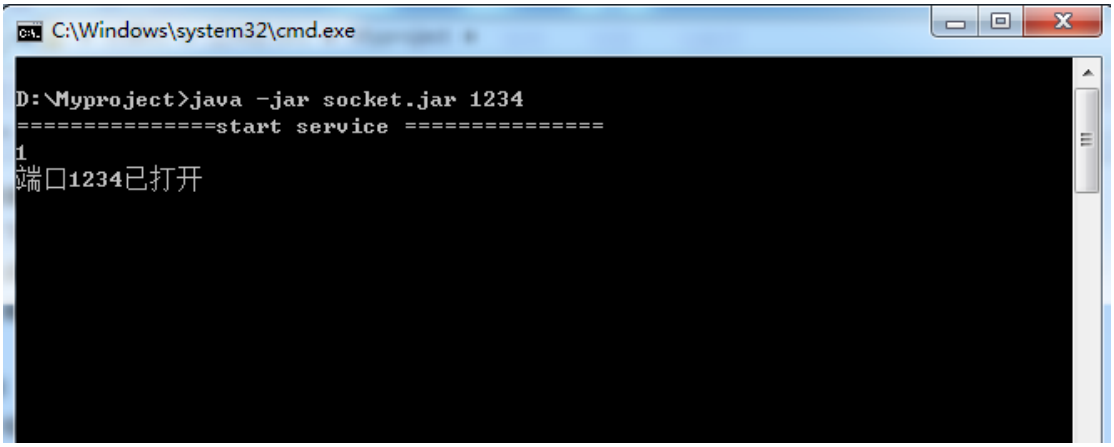
```
                        InputStream in_tmp=socket.getInputStream();
                        OutputStream out_tmp=socket.getOutputStream();
                        while((recvMsgSize=in_tmp.read(receiveBuf))!=-1){
                        String rec=new String(receiveBuf,"utf-8");
                        logger.info(" 收 到 客 户 端 "+adress+" 的 第 "+i+" 次 报 文 为 :
\n"+rec+"\n============================["+recvMsgSize+"]");
                        i++;
        //              if(rec.substring(37, 40).equals("201"))
        //              {
        //                  System.out.println("eeeeeeee");//判断收到报文中的某个字段
        //              //Thread.sleep(500);//延迟 0.5s
        //                  out_tmp.wrITe(sendbuf_tmp);
        //              }else
                            out_tmp.wrITe(sendbuf_tmp);

                    }
                        System.out.println("客户端"+adress+"断开连接");
                } catch (IOException e1) {
                    System.out.println("抛出异常");
                    e1.printStackTrace();
                    return;
                } catch (Exception e){
                    return;
                }
            }
        }

        public static void main(String[] args) {
            System.out.println("===============start service ===============");
            System.out.println(args.length);
```

```
        //new SocketService(int);
        if (args.length==0)
            new SocketService();
        else
            new SocketService(Integer.parseInt(args[0]));
    }
}
```

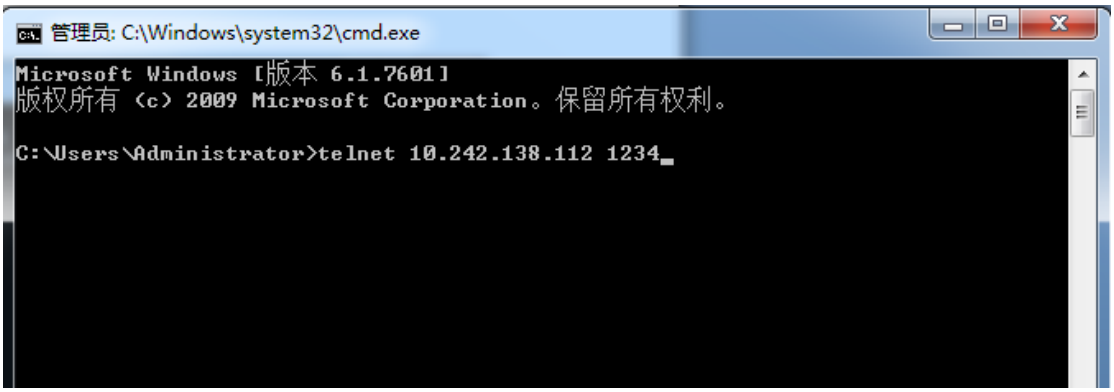
log4j.properties 配置如下:

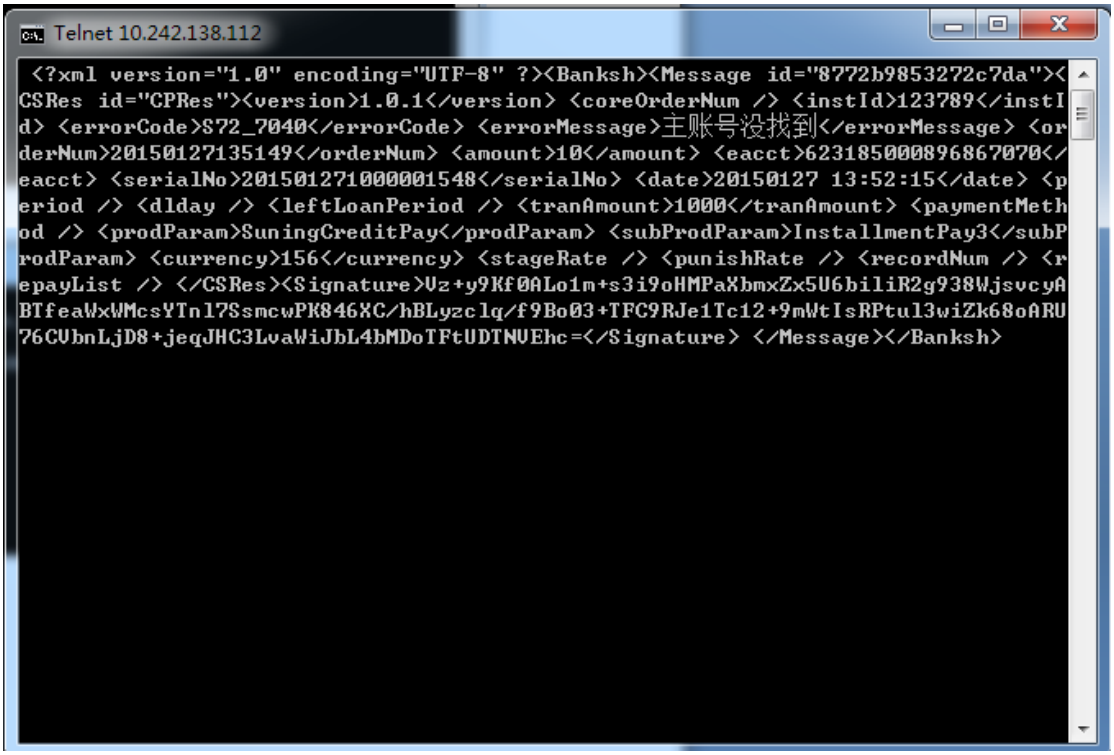
```
log4j.rootLogger=info,appender1,stdout
#log4j.rootLogger=debug,infofile,errorlogfile,D,stdout
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.appender1=org.apache.log4j.FileAppender
log4j.appender.appender1.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
#
log4j.appender.appender1.layout.ConversionPattern=%d{yyyy-MM-dd HH:  mm:  ss }%m%n
log4j.appender.stdout.layout.ConversionPattern=%d{yyyy-MM-dd HH:  mm:  ss }%m%n
log4j.appender.appender1.file=${user.home}/logs/1.log
#
log4j.appender.appender1.Append=false
```

演示图:



这里用 telnet 模拟





经过 LR 场景测试，改挡板 TPS=300 多。

用影响力导图解决问题

译者：于芳

摘要: 你的团队成员有他们不能解决的问题吗？也许是时候尝试下影响力导图方法了。在这篇文章中，留意作者丽莎.克里斯品展示的她怎样使用影响力导图方法解决问题的。影响力导图方法从其他很多头脑风暴和规划工具如思维导图和故事导图中获取很多见解和有用之处。

在他的书影响力导图中，哥吉科.爱德自科解释了一种开发团队和业务商可以共同合作快速识别通往提供最大投资回报比的交付物的路径的方式。 团队们可以建造一个帮助他们快速学习一个特定的方法是否会产生想要的结果，通过回答下面的问题：为什么，谁，怎样和什么来适应难以避免的变化的地图。

影响力导图从其他如思维导图和故事导图的头脑风暴和规划工具里获取很多有用之处。 我发现它给了一个简单的，结构化的方式来让你将焦点聚焦在你试图要规划什么这个意图上。

我一直在试验将影响力导图调试到帮助软件团队成员用他们最想要解决的敏捷软件测试中的问题，在大脑里形成小的试验来改善这些问题区域上的不足。目前为止我的经验里，创建影响力导图正证明为一个解决问题的好方法。

在这个两部分系列的第一部分里，我会就怎样创建一个影响力地图给一个概述。在第二部分，我会帮你整理你自己的影响力导图商店，添加额外的例子和指令。

我推荐读哥吉科的书--它短小精悍，而且大部分内容是图片，而且他的方法很简单；只需要练习几次就能使用。这里是我在工作店铺里已经做的，参与者们已经想到他们在其他情况下不会想到的有创造力的想法。

为什么？

我想这是一个人做事的倾向，先在头脑中对要做什么有个想法再启动寻求解决方案。这也适用于软件的功能的开发。我们的承包商来到我们身边对我们说，“给我们一个具有 X，Y 和 Z 功能的用户界面。”他们经常想给我们实现方法，尽管事实上他们雇佣我们来是因为我们是那个知道怎样开发软件的人。

当客户这么做时，问他们这些问题是很重要的：“为什么你想要这个功能？你想要解决什业务问题？他会添加什么价值？我们会怎样衡量在我们发布生产以后他是否成功？”一旦我们了解了这些意图，我们可以用我们自己的技术和域知识以及我们的创造力来想出最简单最有效的解决方案。

当走到怎样改善我们开发的软件的问题时，包括我们怎样测试和怎样编码，同样的方法也适用。抵制住用交付物开始的冲动问你自己，“我们为什么开这个会议？我们希望达成什么结果？”

在他的书中，哥吉科建议使用“SMART”目标法来回答“为什么”的问题。这是一个已经经过验证是真实的方法，没有新的东西---但是我们有多常想起让我们的目标“具体化，可量化，以行动为中心，实际和及时”？（据大百科全书，对此的确有几个可接受的变量）。对我来说，思考怎样衡量成功是关键。而有一点是很重要的，即当我们尝试实现改善一个问题的路径之后要记得做那个衡量的事情。

在带有粘胶的便条上写下每个目标，然后将他们放在你们的小组工作周围的桌子或墙上的空间里。

这有一个看“为什么”这个问题的一个例子。我的团队在为一个特定的叙事诗交付的用户故事，但是他们中的大多数被拒绝多次。开发者没有去收集有关该故事的全面和正确的需求，因此搅乱不断发生。我的意思是去说，“我们需要做 ATDD 以让开发们更好地理解需求。”但是这是从问题“什么”开始的。那最好去设置一个这样的目标，如“让我们在接下来的 2 个月里将被拒的故事减少百分之二十。”一旦开始，这个过程会产生很多要改善的目标。

谁？

创建一个影响力导图的下一个步骤是思考谁会帮我们达成我们的目标以及谁会跟我们同路。有没有团队外的人可以帮忙？回到我的例子目标，减少被拒绝的故事数量，也许我们需要通过让持续集成版本更快地实现来缩短我们的回馈循环，

但是我们缺少必要的硬件。我们也许需要我们的首席财务官的批准才能拿到为此做预算的钱。

考虑一下那些你平常不会与之合作的人。 做营销工作的同事可以给我们很多关于什么对顾客最重要的信息，这些可以帮助我们恰当地计划我们的时间。顾客的支持和操

作也包含对我们有价值的信息。有时人们甚至不知道我们需要帮助。我记得当我们的团队想到这个做一个妨碍代办事项的想法时，我们将“放慢测试环境”放在首位。我们的数据库管理员和系统管理甚至没有觉察到我们的问题，然后他们才快速发现并为我们创建了新的、运行更快的服务器。

哥吉科推荐填写这个模版：<某人>可以帮助我们达成我们的目标通过<做些不同的事情>。让这个<某人>具体化。想一下谁限制了你的选择。有人能阻止你开始的团队吗？

在便条上写下每个“谁”，然后将他们安排在你的目标周围。这是你影响力导图的第二个层级。

怎样？

对每个你识别出来是帮手或绊脚石的“谁”，明确每个人的行为应当怎样变化，他可以怎样帮助你达成目标，以及他会怎样成为一个阻碍。这些是影响力---想要的会帮助你的团队达成目标的变化。如果我们已经识别到首席财务官作为某个可以帮助我们的人，批准一项采购或者将钱放在我们需要的预算里就是那个影响力。从营销同事或者顾客支持里取得帮助也许跟建立一个跟他们结对的会议或日程一样简单。

你的影响力导图会帮你看到谁创建了一个影响力，以及它会怎样贡献于你的目标。你也许不会选择与所有你辨识出来的演员一起工作也不会让他们做所有的影响力工作，但是那个地图帮你看到所有朝向达成你目标的可变路径。这些为逐步改善你的问题域的小试验们提供了基础。

什么？

在哥吉科的影响力地图中，第三个层级回答了这个问题，“作为一个组织或团队我们能做什么来支持所要求的影响？”当你使用一个影响力地图应用到可能的软件产品时，这些是交付物们---功能本身以及机构性的活动。在一个影响力地图上，每个交付物都是在他们应当去支持的影响力背景下的。

使用影响力地图来帮助解决问题，我们的“交付物”是那些我们会尝试会帮我们解决问题的影响力的小试验。如果我们想要首席财务官批准一个采购申请或者预算变更，我们需要将我们的情况集中到一起呈现给她看，拉她为我们出力。对于我的例子，我们也许决定量化我们的技术债我们的“故事拒绝搅乱”的发生，因为钱会跟首席财务官说

话！一个交付物可能是一个让团队学习怎样使用特定的测试框架的必要的培训课程或者预算的时间来与业务分析师合作来获取更完整的需求。

在我自己的有关影响力导图的试验中，我发现这个问题“什么”是最不重要的层级，哥吉科也在他的书中这么标注。想一下那些会帮我们或者阻碍我们的人，以及他们怎样帮我们或阻碍我们是找到解决问题的创造性方法的真正的火花。

图 1. 一个影响力导图的例子

让你的团队集合到一起来讨论哪些问题你需要让他们变小一些。挑一个能导致最大痛苦的（点投票很擅长做此事）问题，讨论为什么你想要解决这个特定的问题，围绕着它写下 SMART 目标，创建你自己的影响力地图。一旦你已添加了你的“为什么”到目

标单中，开孔你的大脑思考谁可能会帮助你达成这个目标或者谁会阻碍你达成这个目标。对第二部分，我们会浏览下怎样做你自己的影响力导图商店来创建你的团队可以尝试在最难妥协的问题上逐步削弱的大量的小型试验。

移动设备自动化测试工具选型

◆ 作者：郝强

一、问题的提出

最近二两年来，一直在从事移动设备的自动化测试工作，可以说小有心得。但最近由于种种原因，面临着对移动设备的自动化测试工具的更换工作。所以，一个问题呈现在面前。我们需要为我们的项目选出一款新的自动化测试工具，具体来说就是我们要选出可以支持 iOS,android,windows phone 甚至是黑莓等设备的自动化测试工具。当然在讲我的案例前，我希望针对我们对工具的选型工作能够对大家有参考作用，以便未来在您可能遇到类似的问题时，也能够有章可循。

日常我们目前公司的 mobile app 基本上实际上是 web based 的 app,所以在此之前我们的所有自动化测试脚本都是建立在基于对 web 自动化测试工作支持较好的 sahi 上。而我们的 mobile automation 则是使用 chrome 浏览器来模拟 mobile client，使用 sahi 来进行自动化工作，整体来讲，mobile 应用的自动化工具是很成功的，自动化覆盖率比较好，而且运行的速度较快，也为公司节省了许多金钱。但实际上我们也面临一个问题，就是我最终发布的应用实际上是一个 hybrid 应用，而且在实际测试工程中，手工测试人员由于觉得大部分自动化测试跑在 chrome 上，他们对自动化的信任度也不能够达到较高的程度，所以他们觉得，有必要把自动化测试覆盖掉的用例也要手工跑一下，以免出现意外。当然我们非常认可同事的认真工作的态度，但我们确实也应该解决这一问题，让 automation 真正跑在设备上。基于以上，我们打算更换一款工具，能够让我们的自动化测试跑在真实设备上，而非模拟器上。

那么现在问题来了，我们选什么呢？所以我们得先分析一下需求。首先，我们的自动化必须能够跑在设备上。其次，工具必须能够支持 iOS,android,windows phone，blackberry 等。除这两项硬性规定外，我们的需要工作尽可能好用，能够支持企业级应

用，当然如果能够免费最好。

二、 找到至少两组备选方案

首先我们得借助一下 google,百度，查询一下有没有适合的工具。

首先映入眼帘的是它。

我们先来看一下 appium 能干什么？看简要说明它主要能够实现 iOS，Android 以及 FireFox OS 设备的自动化测试，包括 native,hybrid 及手机 web 应用。最重要的是它还是开源的。Appium 是跨平台的，即你可以写一套测试脚本同时运行在 android 及 iOS 平台上。Appium 是基于客户端/服务器架构，它实际上是提供一套 Restful API.它从客户端接收连接，侦听命令，然后在移动设备上执行命令。在客户端我们可以使用任何语言来编写测试脚本。服务端可以运行在不同的机器上。

我们再来看看这个家伙，叫做 SeeTest automation:

SeeTest automation 支持 iOS,Android, Blackberry 及 windowsphone 的自动化测试。它能够在真实设备上录制也可以在模拟器上录制，可同时在不同的设备运行测试，插件丰富，包括 HP UFT(QTP),WebDriver(selenium),JunIT,微软 visual studio 和 pathon。可以与 ALM 连接，Jenkins 和其它持续集成工具。可以通过 usb 或是无线网线与设备进行联接。

这个 SeeTest automation 功能还是真奇全，但这个软件是商业软件，它还有配套的 SeeTest Could 以及虚拟化软件。价格比较昂贵，功能强劲。

我们再在看一款，叫 KeyNote DeviceAnyWhere,这是一款和 SeeTest Automation 差不多的软件，支持支持 iOS,Android, Blackberry 及 windowsphone 的自动化测试。而且它也对设备提供云的支持。功能强，价格贵，是一款企业级的商业软件。

三、对比

通过对备选软件的对比，相信很容易能够得出您所需要的适合的移动设备自动化测试工具。可能也有看官在好奇我们最终选了什么软件。我可以负责任的告诉大家，做为一家有钱任性的大公司，我们毫不犹豫的选择了商业软件，而且那两款商用软件都有啊。

抛开钱的问题，做为一家企业，你有可能会有这样的潜在需求，即你希望你的设备是可以集中管理的，比如说，我在大连有一个移动设备中心，然后在北京也有一个，在上海也有一个，如果贵公司是全球企业，可能你在美国，英国等都有一个移动设备中心，如果需要设备是共享的，那么如果你选择的自动化测试工具提供云的支持，那么每个中心的设备大家都是共享可用的，你也可以最大化的利用到每一台设备。

当然，大多数互联网公司 我相信会更可能选择开源的 appium，它免费，又可以自己动手定制，一般来讲中国大多数互联网公司对自动化测试云的要求不是那么强烈，如果真有要求，我们一般也会自己想办公，利用现有的开源软件及技术，自己来搭建。

讲到这里，实际上我们应该已经了解到，对于任何一种工具的选择，我们首要考虑其功能性，看能够满足我们的要求。其它，看价格。性价比高的优先考虑。最后我们一定将潜在的需求也列出来，是否有设备云的要求，是否有持续集成的要求，是否对我们所擅长的编程语言有支持，以及是否有利用扩展等。

除此外，如果您打算入手商业工具，这里还建议各们先试用，小范围使用评估并反馈结果。一般商业软件都有试用期，通常为一个月，如果试用一个月不够，通常情况下要求延长试用期也是可行的。

测试女巫之石头变宝石篇

作者：妞妞

摘要： 将统计学_6 sigma 应用到测试管理中，需要使用的 6 sigma 工具如下：
DOE，主效应分析；柏拉图分析

一、前言：

测试女巫又来啦，大家还记的吗？前三次我们主要以“找到 bug 产生的原因”以及“提高 bug 产出效率”和“通过自动化节省人力”为例介绍 DOE，柏拉图，主效应，交互作用，相关性，鱼骨图，Xbar Chart，One Way ANOVA，Two Way ANOVA，：QFD，DFMEA，量测系统，Johnson Transformation 这些方法，并着重介绍如何将这些方法应用到我们软件测试中。

我们在介绍上述工具时，共性是都用了一个比较完整的 DMADV 的流程改进方法来对我们的工作流程进行改进，但是在实际工作中可能我只需要对一个比较简单的事情进行分析，只需要用到几个简单的 6 sigma 的工具，用不到 DMADV 这样“高大上”的流程，那该怎么应用呢？是不是所有的 6 sigma 分析都要使用这样的流程呢。答案当然是否定的！因为对于 6 sigma 来说最核心的是一种思维模式，虽然它提供了很多工具，但是思维才是最重要的，工具只是辅助而已。

OK，我们来列举一下在实际工作中遇到的问题：

作为测试部门不可避免的要与其它部门的人员合作，例如项目经理部门，软件开发部门……在项目开发的过程中，掌控项目进程的主要是项目经理，但是，他们提出的需求都是合理的吗？因为他们提出的错误需求，浪费了我们多少人力？根据笔者的经验，我们测试部门的 Project leader 一直在抱怨这个问题，但是我发现仅仅是抱怨而已，下一个项目，同样的问题依然惊人的重复着，难道对于这样的问题，我们不能做点什么吗？

随着年龄的增长，我们经历的项目越来越多，如果我们注意对这些项目进行有目的地收集资料，以及使用 6 sigma 的思维和使用 6 sigma 相关的工具将这些资料进行专业的

分析，使之真正成为我们的经验，就会让我们的“核心竞争力”不断加强。打一个容易理解的例子就是将“一堆平淡无奇的石头”变成“一堆闪闪发亮的宝石”。让我们从现在开始就培养这样的习惯吧，即从做的每一个项目中，反复“压榨”，不断抽离出宝贵的可量化的经验，如此下去，社会怎么会“舍得淘汰”我们这种优秀人才呢？哇哈哈想起来顿时有了很大的干劲！

好的，既然有想法就要立刻开始着手去做！这次我们主要用到的工具为：DOE，主效应，柏拉图。在第三十四期的杂志中已经介绍了这些工具的原理和用法，此份文档中也会用到这些工具，因为之前已经介绍过了，这里就不再赘述。

二、6 sigma 常用工具基础知识介绍

DOE 实验设计方法

此工具已经在第 34 期“51 测试天地杂志”中的“测试女巫”此文章中有详细的介绍，这里就不再赘述。

主效应分析方法

此工具已经在第 34 期“51 测试天地杂志”中的“测试女巫”此文章中有详细的介绍，这里就不再赘述。

柏拉图分析方法

此工具已经在第 34 期“51 测试天地杂志”中的“测试女巫”此文章中有详细的介绍，这里就不再赘述。

三、应用到实际工作中_总结工作篇

我们都知道如果技术经理和项目经理等这些前期与客户谈需求的“核心人员”在前期工作没有做好，此项目在真正实施阶段，会遇到很多本应该可以避免的问题，例如笔者近期做的一个项目就是一个非常好的案例。某一个项目，软件开发和验证阶段被压缩到一个正常项目时间的一半不到的时间。在项目真正执行阶段，所有的测试都被要求提前，给我们造成很大的困扰，针对此问题，对于测试影响最大的项目：流量吞吐量的问题，所以进行在真正执行测试前进行的设计如下

1. 确认因子

确认影响“流量吞吐量”以及“花费人力”的因子（即将实际的原因翻译成 6 sigma

语言，这个步骤非常重要，这个步骤的本质就是将我们的思维转换为 6 sigma 思维，然后通过这些思维搜集数据）

1) 项目所处的阶段

因为在不同的项目阶段，软件以及硬件的品质是不一样的。我们以 Demo Alpha Beta Release 来将一个项目定义为 4 个阶段，即从 6 sigma 的角度来考虑就是有 Project phase 此因子有 4 个 levels。

2) Test sITe

不同的测试实验室，对测试结果和花费的人力是否有影响，目前我们公司对于吞吐量的测试有两个实验室。即从 6 sigma 的角度来考虑就是 Test sITe 此因子有 2 个 levels。

3) Tester

不同的测试人员，对测试结果和花费的人力是否有影响，目前我们从 6 sigma 的角度来考虑就是 Tester 此因子有 2 个 Levels。

4) Human cost

此轮测试花费多少人力，以多少人/天来计算，一天的时间为工作时间 8 个小时.如果结果存在小数部分则意味着此时间为几天+几个小时。

5) Result

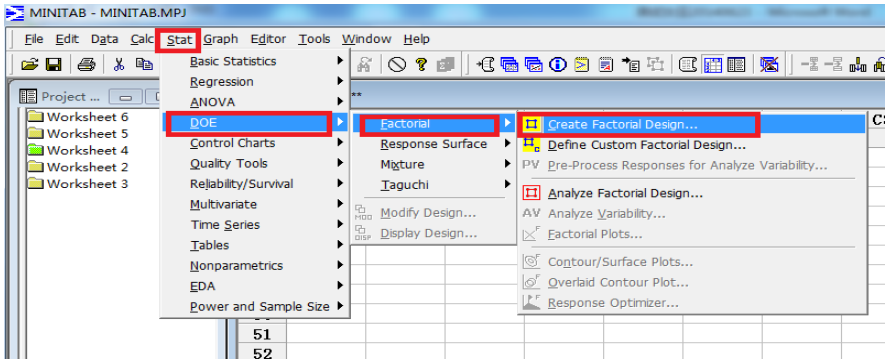
此次测试结果经厘清是否符合测试预期，例如对于“流量吞吐量”的测试。我们假设的前提条件是待测物的硬件天线性能良好，天线的所有实验室指标均达到 3GPP 要求；天线性能良好，天线性能所有实验室指标均达到 3GPP 要求。软件搭配硬件的整体稳定性较强。但是在发现测试结果为 Fail 时，经厘清，最后发现前提条件并不成立，此时测试结果为 0；反之则为 1。

2. 实验方案设计

1) 根据我在第 34 期“51 测试天地杂志”中的“测试女巫”此文章中教给大家的 DOE 实验方法，先对上述的因子即 X 和结果 Y 进行实验设计（即根据翻译完毕的 6 sigma 语言，进行实验设计，并记录结果），

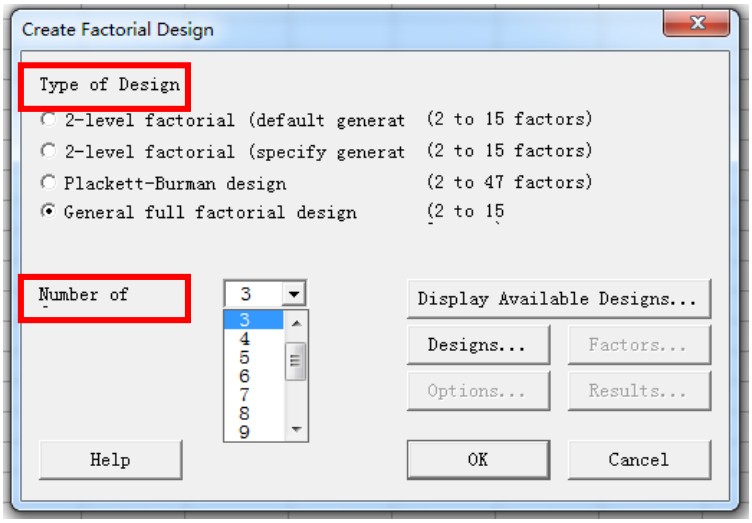
步骤如下:

a. 选择 Stat->DOE->Factorial->Create Factorial Design 如下【图 1】



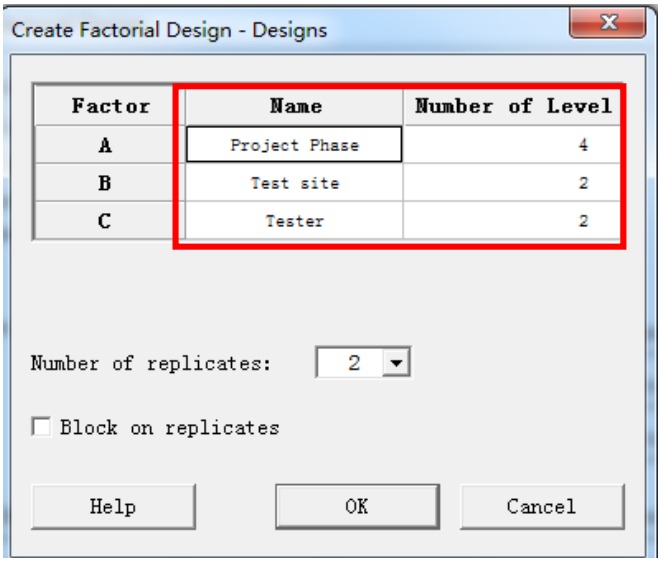
【图 1】

b. 选择实验设计的总类为 General Full Factorial Design（为什么选择此种设计类型，请参考第 32 期“51 测试天地杂志”中的“测试女巫”此文章中有关“减肥”的分析说明），因为我们在确认影响两个 Y（测试结果和 Human Cost）的因子时，找出 3 个因子，所以 Number of factors 选择 3。如【图 2】



【图 2】

c. 我们在确认因子时同时也把每个因子对应的水准确定下来了，点击 Design 将因子和水准填到如下【图 3】的位置。



【图 3】

d. 点击 OK 后会自动产生一个实验设计图标如【图 4】

StdOrder	RunOrder	PtType	Blocks	Project Phase	Test site	Tester
11	1	1	1	Beta	2	1
24	2	1	1	Alpha	2	2
19	3	1	1	Demo	2	1
32	4	1	1	Release	1	2
9	5	1	1	Beta	1	1
12	6	1	1	Beta	2	2
16	7	1	1	Release	2	2
8	8	1	1	Alpha	2	2
20	9	1	1	Demo	2	2
28	10	1	1	Beta	2	2
30	11	1	1	Release	1	2
22	12	1	1	Alpha	1	2
23	13	1	1	Alpha	2	1
2	14	1	1	Demo	1	2
10	15	1	1	Beta	1	2
1	16	1	1	Demo	1	1
3	17	1	1	Demo	2	1
13	18	1	1	Release	1	1
29	19	1	1	Release	1	1
25	20	1	1	Beta	1	1
18	21	1	1	Demo	1	2
27	22	1	1	Beta	2	1
4	23	1	1	Demo	2	2
21	24	1	1	Alpha	1	1
7	25	1	1	Alpha	2	1
17	26	1	1	Demo	1	1

【图 4】

f. 根据在整个项目中执行“流量吞吐量”的条件，以及当时的结果:

包括 Result 以及 Human cost 来记录数据如下【图 5】

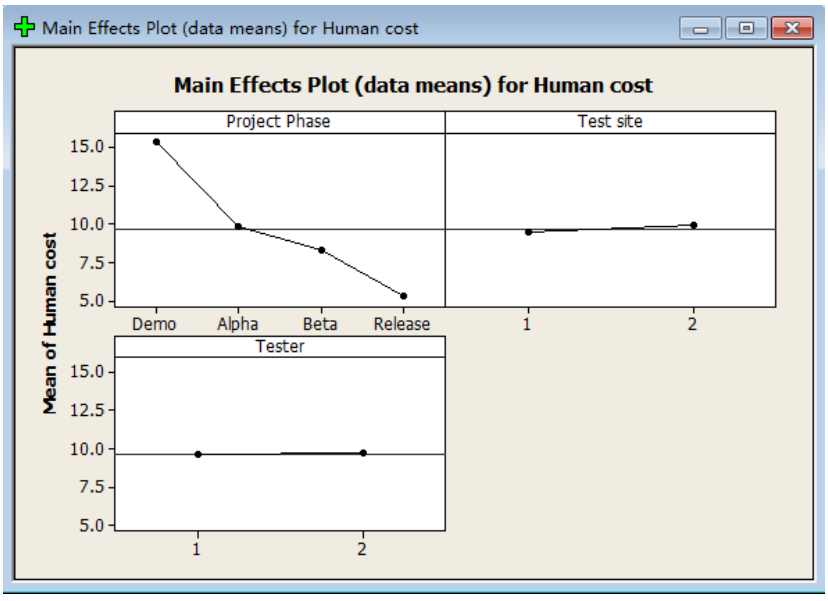
StdOrder	RunOrder	PtType	Blocks	Project Phase	Test site	Tester	Human cost	Result
11	1	1	1	Beta	2	1	8.0	1
24	2	1	1	Alpha	2	2	10.0	0
19	3	1	1	Demo	2	1	15.0	0
32	4	1	1	Release	1	2	5.0	1
9	5	1	1	Beta	1	1	7.5	0
12	6	1	1	Beta	2	2	8.2	1
16	7	1	1	Release	2	2	5.5	1
8	8	1	1	Alpha	2	2	9.5	1
20	9	1	1	Demo	2	2	15.5	0
28	10	1	1	Beta	2	2	8.5	1
30	11	1	1	Release	1	2	6.0	1
22	12	1	1	Alpha	1	2	10.5	1
23	13	1	1	Alpha	2	1	9.0	0
2	14	1	1	Demo	1	2	14.5	0
10	15	1	1	Beta	1	2	8.0	1
1	16	1	1	Demo	1	1	15.8	0
3	17	1	1	Demo	2	1	16.0	0
13	18	1	1	Release	1	1	4.6	1
29	19	1	1	Release	1	1	5.5	1
25	20	1	1	Beta	1	1	8.5	1
18	21	1	1	Demo	1	2	15.3	0
27	22	1	1	Beta	2	1	8.2	1
4	23	1	1	Demo	2	2	14.4	1
21	24	1	1	Alpha	1	1	10.0	1
7	25	1	1	Alpha	2	1	10.5	1
17	26	1	1	Demo	1	1	16.0	0

【图 5】

3. 数据分析

1) 先对 Human cost 进行分析

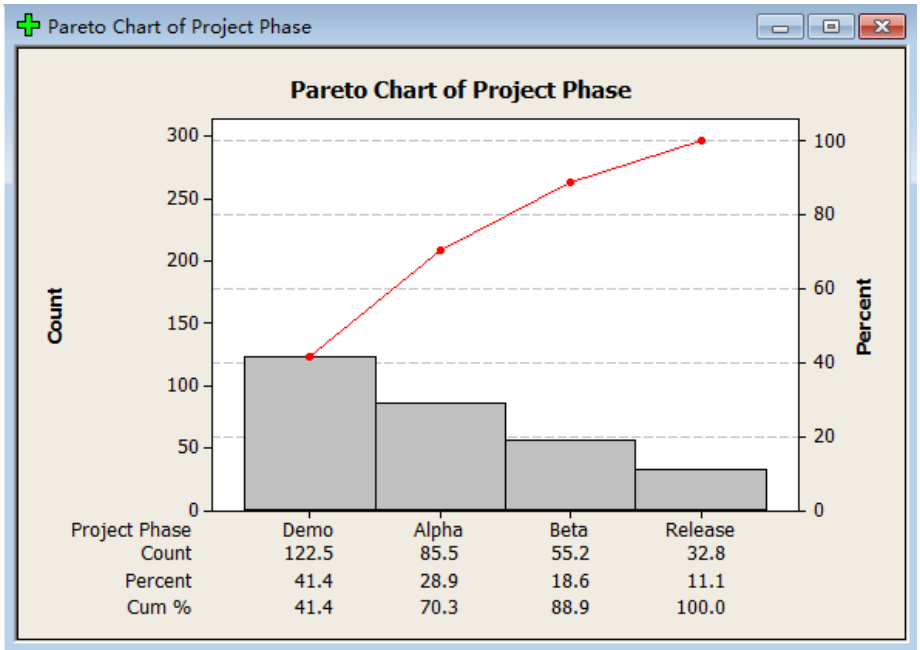
a. 先进行我们最关心的主效应分析如【图 6】(主效应的原理以及如何选择主效应请参考第 34 期“51 测试天地杂志”中的有关主效应的解释以及相关例子。)



【图 6】

根据上述分析可以看出 Project Phase 对于人力的花费是非常明显的影响。

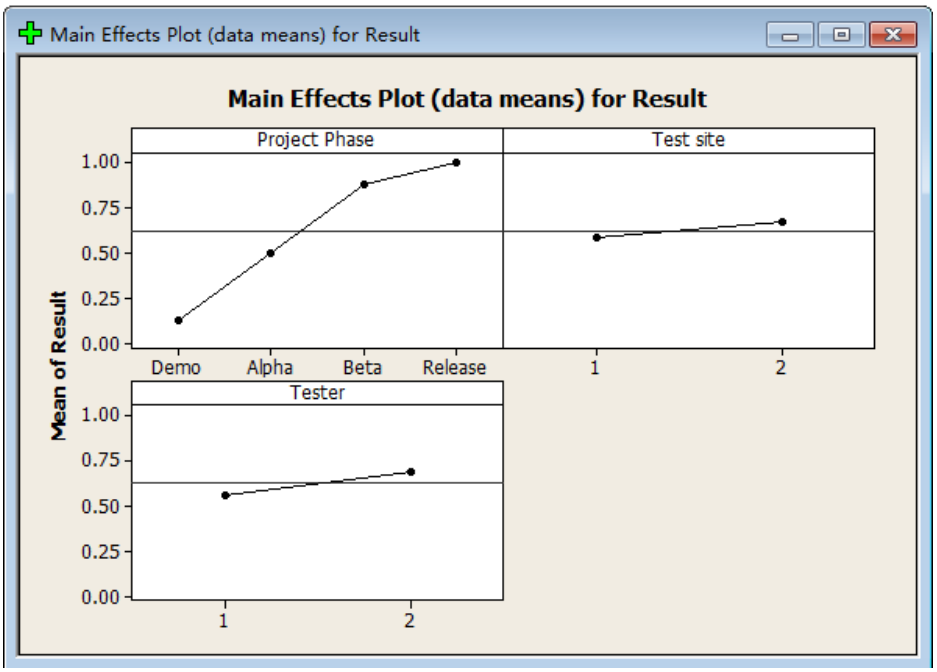
b. 我们进一步分析 Project 的各个水准对于人力花费的影响，如下【图 7】，可以看到我们人力的 70%都是花在 Demo+Alpha 这个阶段，只有 30%花在 Beta 和 Release 阶段



【图 7】

2) 对 Result 进行分析

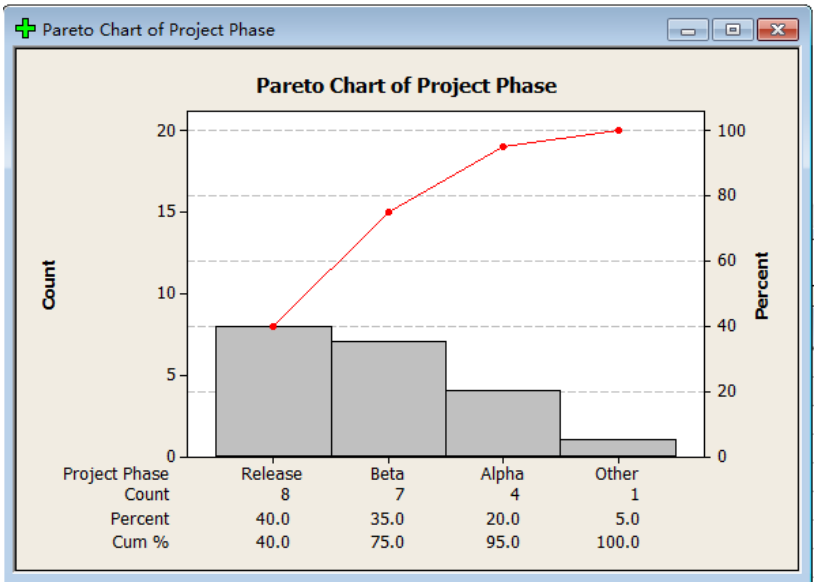
a. 主效应的分析如【图 8】。对于测试结果是否符合预期，也是 Project phase 对于 Result 有非常明显的影响



【图 8】

b.我们进一步分析 Project 的各个水准对于 Result 的影响。

如【图 9】 从图中可以看出结果有效的比率 80%以上集中在 Release 以及 Beta 阶段；且 Release、Beta、Alpha 的结果有效的比率达到了 95%!



【图 9】

结论

结合对 Human Cost 以及 Result 的分析，可以得到如下结论:

对于这个“非常特殊的项目”我们花费了大于 70%的人力做了只有 25%有价值的工作!!! 多么触目惊心的结论!! 我们在前期(Demo+Alpha)的工作做得兢兢业业，经常加班，但是因为按照做项目的规律这个阶段的产品并未达到测试“流量吞吐量”的标准，由于项目时间紧张，项目经理和技术经理不顾这个基本规律，盲目强势要求测试这个项目，最终的结果就是这样!! 花费大量人力，且没有任何实际意义的产出，这是多么大的浪费!!

分析到这里，作为测试部门的主管，该有多大的气愤！但是我们还是需要解决问题不是吗？好的，冷静一下，我们继续分析，继续分析到底是什么原因导致在 Demo 和 Alpha 阶段“流量吞吐量”的测试结果无效呢，还有到底是什么原因导致在这两个阶段花费我们这么多的人力呢？我们进入下一阶段的分析!

对于导致“流量吞吐量”此项目测试结果的无效以及为什么花费这样多的人力的分析。

进一步分析

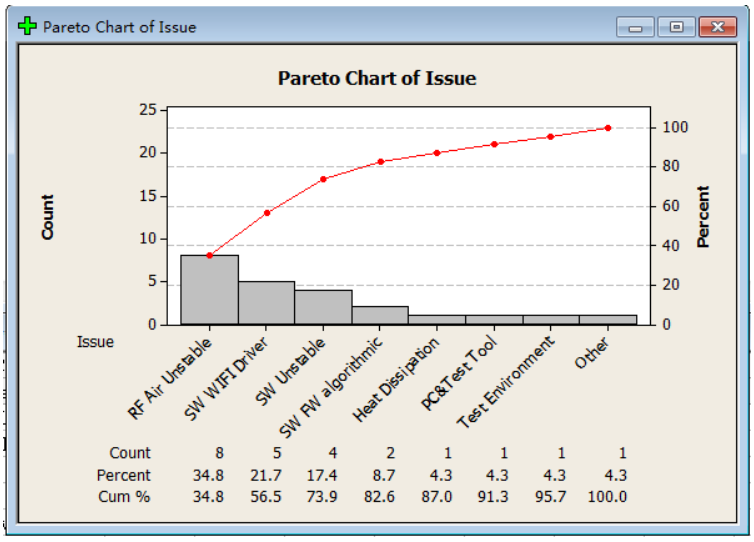
1) 我们将每一次测试无效的原因做了记录如下【图 10】，其中测试人员分为测试人员 1 和测试人员 2

Issue	Time	Tester
RF Air Unstable	7	1
SW FW algorithmic	2	2
SW WIFI Driver	5	1
PC&Test Tool	1	2
Test Environment	1	1
Test Method	1	1
SW Unstable	4	2
Heat Dissipation	1	2

【图 10】

2) 对 Issue 的失败次数进行柏拉图分析，如下【图 11】

可以得到：Air 天线的稳定性，SW WIFI Driver，SW 的稳定性，SW 软件的算法以及，散热问题这些问题共占所有测试无效的比率为 80%!! 这些问题都是说明不管是软件还是硬件的设计都没达“流量吞吐量”的测试标准!!



【图 11】

改进措施

我们已经知道比较确切的原因了，虽然我们将原本在后期的测试提到项目前期进行，对这个项目并没有实际的意义！因为由于软件或者硬件的不稳定导致我们测试的结

果 75%没有价值!!

接下来，作为测试主管，我就可以平静地写出一份有理有据的报告，告诉这些“核心人员”，由于他们的错误决定，导致我多少人力白白的浪费。

所以我有充足的理由拒绝他们以后的项目类似的需求!!

是不是做上述的改进就够了呢，当然不行，因为我们必须研究出一个替代方案，即如果老板或者客户非常想知道在项目的前期阶段“流量吞吐量”是什么状态，我只能“照本宣科”得安排人力进行“流量吞吐量”的测试吗？当然不是，针对此问题我研究出一个简便的测试方案：这个方案可以大大减少我们的测试时间以及测试人力。如果在非常状态下，必须要测试“流量吞吐量”的测试，我们可以优先安排这个替代方案的测试，如果替代方案不通过就没必要安排“流量吞吐量”的正式测试了。

五、总结

我们在前三期介绍了很多 6 sigma 的工具，并讲解了它们的应用，但是大家发现没有，这次我们并没有介绍新工具，难道是 6 sigma 没有新工具了吗，当然不是，是我觉得应该引导大家多思考我们学了这么多的工具，该怎么用？这一期我们使用的工具非常简单，但是这一期的思维是非常珍贵的，作为测试部门，在我们公司是属于“弱势群体”，是属于“人微言轻”的部门，我很明白这一点，我曾经抱怨过，放弃过，但是我慢慢发现抱怨是没有用的，如果我想改变些什么即我想打破目前现有的流程（在我看来很不合理的流程），首先我就应该给出一份有理有据的无可辩驳的报告，让他们看到，目前的流程存在的问题在哪里，然后才有可能做一些改变！所以你目前所处地位并不重要，因为“正确的思维方向+不断进取的心+不断学习新的知识+不断将学习的新知识应用到实际”才最终能决定你的人生走向!!

我们来总结一下这次的非常重要的思维问题:

因为我们“人微言轻”，对于那些“核心人员”不合理的要求，我只能 Follow，客户最大嘛！虽然开发验证的时间不断的压缩我也只能接受，但是对于当前这个项目我可以接受，但是不是没有思想的去全盘接受，我可以将我认为不合理的需求（虽然在当时没有任何人能听取我的意见），翻译成 6 sigma 的语言，在真正执行测试时，收集测试的各项数据，在项目结束后，我可以对这些数据进行 6 sigma 的分析，真是不分析不知道，结果比我们感觉到的现状还要惊人!! 所以在平时工作前，你要首先想想，我觉得

哪部分工作存在需要改善到的部分，那我对于这些需要改善的部分需要先做一个“语言翻译”工作，即将我们人类的语言翻译成“6 sigma”的语言，即你需要观察的产出是什么，因子是什么，每个因子的水准是什么。语言翻译好了，然后就可以进行“实验设计”了，然后就可以开始我们平时的测试工作，将我们想要记录的数据记录下来。然后积累到这个项目结束，我们就可以对这些数据进行 6 sigma 的分析了!! 相信我，分析的结果往往会让你大吃一惊!!

当然对于 6 sigma 不仅可以帮助我们如何应付那些强势的部门的无理要求，而且还可以帮助我们总结一个芯片的特点，因为就我们公司而言，一款芯片可以应用到很多项目中，如果我根据一个项目的数据，可以总结出一款芯片的特点，对于其它项目的测试甚至开发，是不是有很重要的借鉴作用呢？？

还是那句话：对于 6 sigma 可以带来的奇妙旅程，我将不懈的坚持探索，只有不断的使用它们才觉得原来很多很有深度的内容并没有彻底的理解，所以“路漫漫其修远兮，吾将充满欢喜的上下而求索”!

一个测试者眼中的敏捷和 Scrum 方法

◆译者：于芳

摘要：敏捷软件开发模式已经跨过创新阶段，快速跨进早期采用阶段。你注意到每个所见之处都有提到敏捷和 Scrum 技术吗？这篇短文将描述关键的敏捷/Scrum 概念，在一个敏捷项目里使用 Scrum 管理的不同阶段的情况和作为一个质量保证工程师或测试专业人员应当考虑的首要的三件事。如果你的机构公司正在关注敏捷/Scrum，或者你想要跟上行业最新趋势，请接着读。

敏捷软件开发方法是一种将增加的功能通过较小的循环逐步添加到项目中，工作是由自我组织的团队以高效合作的方式拥抱和适应变化来保证客户需求被真正满足的方式来完成软件开发项目的方法。敏捷软件开发方式不是新生事物，事实上在 20 世纪 90 年代他就被引进作为一种减少成本，最小化风险和确保最终产品是客户真正需求的方式来使用了。敏捷方法背后的思想是取代创建功能巨大的版本（常常延后与市场需求），一个机构可以通过将每个版本分裂成更小更短的 1 到 6 周的循环来适应快速变化的要求和条件。每个循环被称作一个迭代，或者冲刺，而这几乎就像项目本身的一个迷你小型软件项目，因为他包含所有发布增加的新功能的必要的项目功能任务。在理论上，每个冲刺结尾，产品应当被备好做一次总版集成。敏捷方法强调实时沟通，相比较书面文档和生硬的流程沟通，更偏好面对面的沟通。除此之外，敏捷流程方法引进了一个广泛使用的技术之一---将产品需求以讲用户故事的形式表达出来。每个用户故事都有各种各样的字段如“主角”，“目标”或者他们需要执行的一个任务，一个解释。

大多数敏捷团队版扩所有发布软件的必要人员。最少程度上，这要包括程序员和他们为之开发程序的组或团队，通常被称作他们的

“顾客”（顾客是定义产品的人，他们可能是产品经理，业务分析师或者实际的顾客）。一个典型的敏捷团队还将会包括一名 Scrum 大师，测试工程师，交互设计师，技术作者和经理。

什么是 Scrum？Scrum 是一个真正的能够便捷敏捷开发软件的项目管理方法，他使得自我组织的敏捷团队诞生。

一名 Scrum 大师就像一个传统的项目经理，他/她俯瞰团队沟通，需求，时间安排日

程和项目进度的中心。但是它也是跟传统项目经理很不一样的，因为他/她的主要责任在让团队之间的沟通更简单便利，同时提供指导和教练，去除阻碍团队能力发挥的障碍来达成目标。不像传统的项目经理，Scrum 大师不主管团队，因为一个敏捷团队是建立在这样一种哲学思想上---团队成员与其他团队成员之间互有承诺，而不是对管理权威。

使用 Scrum 敏捷开发项目的各阶段

敏捷可以定制为就大小，迭代时间，经验等要求而适应每个公司的方法，但是典型的敏捷项目会有这些阶段和里程碑。

1、启动会议。尽管这看起来对任何项目都是例行事务，对敏捷开发项目，这是一个让项目运行的关键要素。这个启动会议的目标是让团队里的每个成员一起来评审产品备份单（即产品拥有者在用户故事形式里起草的产品需要的所有需求的清单），和用户个人习惯偏好（或者叫每类产品用户的描述）。在我看来，这种方式更友好也更清楚地介绍产品需求，因为你真的能在一开始就能对谁在使用这个产品，他们试图达成什么以及为什么这样做有更多的预见能力。一个启动会议通常持续至少半天时间，每个成员一起浏览一个“故事编写工作商店”--在这里选择故事然后将之解构成克编程的任务，与时间估算放在一起写进一个白板里以完成。如果你从来没见过产品积压代办事项，可以在 QAZone 这里看几个例子。有时候真正的顾客会被邀请来参加启动会议，和敏捷团队一起评审和搞清楚产品的代办事项。

2、冲刺/迭代规划的下一个步骤，是团队共同决定冲刺目标和冲刺的代办事项（为那个特定的冲刺按照优先顺序排列要做的的工作清单）。在团队成员共同创建冲刺代办事项时，需要将故事分成子故事或较小的任务。在这种集体团队活动中，你真的可以看到项目管理的差别（至少如果你使用比较生硬和正式的瀑布模型如背景模型时会看到），因为这里没有管理权威分配任务给团队成员。在一个敏捷团队里，所有成员联合将困难级别连写到特定的任务上，他们可以移除故事和/或任务，也可添加额外的故事和/或任务，而任务基于自愿基础在团队里分配。不像传统的项目经理功能，这个会议里的 Scrum 大师角色是基于会议中团队反馈和共识维护产品代办事项，确保每个成员在自愿基础上拿到的任务没有超负荷，同时简化成员对团队效力的流程。

3、既然冲刺规划工作在冲刺代办事项形式下准备好了---这个过程是动态的，非固定的，事实上很可能他会基于新故事，新任务和/或在迭代中发现的阻碍来调整和改变---Scrum 会议会在同时同地每天开设一次。如果你从未参加过 Scrum 会议，这些会议属性

是变化十分快速的，从不超过 30 分钟，理想的为 10 到 15 分钟。目的是来回巡视，如此每个团队成员可以回答 3 个问题：自从上次会议以来我完成了什么（开发了什么，测试过什么，编写过什么故事等），接下来我会要去做什么，以及阻碍我完成我的目标的问题是什么，如果有的话。这些会议很重要，他们确保团队围绕着他们的冲刺目标活动，或者适应/旋转和变化优先级和任务如果遇到新的故事，阻碍或者新场景的需要的话。

4、在冲刺或迭代结尾，通常一个最终验收会议会召开，这主要是展示团队已经完成的东西，向顾客或者更大的观众发送一个产品展示。

5、在一次迭代末，还有一个冲刺回顾会议，类似于其他传统项目的事后讨论会议，这样团队成员聚在一起评估哪些做得好，哪些工作需要在下次迭代中改善。

一名质量保证专业人士应当想到的首要的 3 件事，当一个机构采用敏捷/Scrum 开发技术时，敏捷和 Scrum 真的在改变测试在项目里被认为和看待的方式。测试不是最后的一个阶段；它真的是在集成整个迭代循环中，而且与编程任务肩并肩一同进行。**在我的经验里，当在一个敏捷项目里比较测试角色时，或者当使用那个更死板、正式的方法时，我发现在敏捷方法里是这样:**

1、质量保证和开发团队间更好的沟通和更好的合作。“给我需求”，“我会返回给你缺陷和报告”的日子不复返了，质量保证工程师在项目的一开头就介入---与开发团队同步，而且他们对产品需求和顾客需要的相同信息同时有权限进入。这种从开头进入的参与，结合开发和质量保证工程师现在是相同的敏捷团队的部分，他们每天一起工作，他们对彼此对冲刺整体的成功要执行的任务有完全的可见性的事实，意味着他们之间的更好更频繁的沟通。除此之外，因为整个团队满足了每日（开发，测试，产品管理等等），有更好合作的机会和更多执行某个特定任务的观点。而且传统的你会在 QA 和开发之间发现的“竞争对抗”现象也消除李因为现在是一个单一的敏捷团队工作来达成一个共同的目标。

2、一种新的开发和测试人事之间的“同行对同行”的关系 你应当准备好更多地“说出来”。敏捷方法都是有关组建自我组织的团队，QA 工程师或测试员的话语跟开发人员同等重要。思考一下这种方法。在每日的 Scrum 会议中，每个团队成员问有关他们的成果（测试，开发，写作产品文档等等），将来计划和障碍，对所有成员一视同仁。在敏捷团队里，如“我们将怎样去测试它”的问题跟“我们将怎样去建造它”同等重

要。另外，因为测试人员倾向于特别擅长挖掘需求和辨识不断变换的场景，（尤其是当他们完全可以看到产品需求和用户需要时），他们提供了有价值的设计观点和架构决策，从项目一开始。而这些贡献换来了更多的尊重和他们的开发同僚的欣赏。

3、寻找优化测试工作量的方式将成为一个“必须做的事”你真的需要思考自动化，规划和十分有效地执行你的测试工作。在不超过 6 周的更短的开发周期下，版本一直在发布的方式下，测试的工作量真的需要尽可能地优化，因为没有像这样的单独的测试阶段。 达成这个想法的一个方式是在项目里平衡探索性测试和自动化测试的比重。探索性测试在搜寻缺陷，改善机会和缺失功能上很方便。所以你应当在每个新冲刺开始计划“探索”产品，或者在冲刺周期内产品功能有变更的任意时间里计划“探索”。类似地，你需要计划和构建你的脚本来在冲刺内执行自动化功能和回归测试，因为没有足够的时间来执行充分的手工测试。

要记住的是没有真正冗长的需求文档或说明书---除了那些附录在代办事项文件的故事，因此唯一能够确保每个功能在被称作“完成”之前被产品拥有者完全开发，测试和验收的方式是使用冲刺代办事项作为你自己的测试计划（或者为每个功能写一个测试用例或脚本）。有的团队在把测试用例场景当作需要添加到产品/冲刺代办事项文件以用作规划和跟踪的目的的入口。另外一个要考虑的因素是开发在测试中参与比重更大，因此你应当平衡这个比重，与他们紧密工作来计划和建造更能覆盖显示范围的自动化脚本。

如果你享受参与到产品决策制定中，帮助塑造产品的外形和工作的这种方式，在合作的环境里工作，鼓励团队工作和与你的开发同行建立同行关系，你就会享受在敏捷项目里工作。在下游，敏捷软件开发方式可能在一开始会有点隐晦。敏捷的一切都是关于拥抱和快速适应变化的方法---这可能在一开始很难让人接受，加上有新的流程和新的沟通方式，所以你可能会有些不太愿意使用它。但是，一旦你进入动态地敏捷软件开发循环，那将会是一个充满乐趣和感受到力量的经历！

浅谈软件测试项目的质量保证

◆作者：田辉

摘要： 2014 年 1 月 1 日实施生效的 CNAS CL45-2013 规定实验室管理体系中应包括："测试质量保证要监督项目人员的测试工作、审核测试过程及形成的测试工作产品与标准、规范、过程文件的符合性"，正式将"测试质量保证"引入了第三方软件测评机构的管理体系。相对于软件开发过程的质量保证，软件测试质量保证有其自身的特点。本文从软件测评机构的立场出发，阐述了软件测试作为独立项目时的质量保证过程及其活动。

随着软件在各行各业的日益普及，软件质量问题导致的不良后果也越来越多，软件质量的重要性日益突出。作为保证软件产品质量的最直接最有效的手段，越来越多的企业和用户逐渐意识到软件测试的重要性，第三方软件测试行业应运而生。然而随着软件开发规模的增大、复杂程度的增加，以寻找软件中的缺陷为目的的测试工作就显得更加困难。

统计表明，开发较大规模的软件，有 40%以上的精力是耗费在测试上的，即使富有经验的程序员，也难免在编码中发生错误，何况有些错误在设计甚至分析阶段就已埋下祸根，无论是早期潜伏下来的错误或编码中新引入的错误，若不及时排除，轻者降低软件的可靠性，重者导致整个系统的失败。

为了尽可能多地找出程序中的错误，生产出高质量的软件产品，加强对测试工作的质量保证就显得尤为重要。在第三方软件测评机构中，软件测试作为独立的项目存在。随着第三方软件测评机构所遵循的 CNAS CL45-2013 正式引入“测试质量保证”，第三方软件测评机构的软件测试项目如何开展质量保证日益受关注。

一、软件质量

软件质量是指软件产品的特性可以满足用户的功能、性能需求的能力，它是贯穿软件生命周期的一个极为重要的问题，是软件开发过程中所使用的各种开发技术和验证方法的最终体现，因此，在软件生命期中需要特别重视质量保证活动，以确保生成高质量的软件产品。

与传统意义上的质量概念相比，软件质量是针对软件的某些特性而言。一般认为软

件质量由三部分构成:

（1）软件产品的质量，即满足使用要求的程度。

（2）软件开发过程的质量，即能否满足开发所带来的成本、时间和风险等要求。

（3）软件在其商业环境中所表现的质量。

总体而言，高品质的软件应该是相对无缺陷或只有极少量缺陷的，它能够准时交付给客户，所花费用在预算内，满足客户需求，并且是可维护的。但是，有关质量好坏的最终评价依赖于用户的反馈。软件质量保证就是为了确定、达到和维护需要的软件质量而进行的、所有有计划、有系统的管理活动。

二、软件测试与软件质量保证

软件测试与软件质量保证两者之间既存在包含关系，又存有交叉的关系。测试不是质量保证，二者并不等同。软件测试能够找出软件缺陷并进行修改，确保软件产品满足需求，从而提高软件产品的质量。而软件质量保证则是避免错误以求高质量，并且还有其他方面的措施以保证质量问题。

1. 软件测试与软件质量保证的共同点

（1）软件测试和软件质量保证都是以尽力确保软件产品满足需求为目的，从而提高软件产品的质量。

（2）软件测试和软件质量保证的流程都是贯穿整个软件开发生命周期中。

2. 软件测试与软件质量保证的不同点

软件质量保证侧重对软件开发流程中的各个过程进行管理与控制，杜绝软件缺陷的产生。而软件测试则是对已产生的软件缺陷进行修复。

三、软件测试项目的质量保证过程概要图

软件质量保证的活动主要包括：制定软件质量要求、组织正式审查、软件测试管理、对软件的变更进行控制、对软件质量进行度量、对软件质量情况及时记录和报告。软件质量保证的职能是向管理层提供正确的可行信息，从而促进和辅助设计流程的改进。软件质量保证的职能还包括监督测试流程，这样测试工作就可以被客观地审查和评估，同时也有助于测试流程的改进。

第三方软件测试机构的测试项目大多是客户在完成软件编制之后，以委托的方式交由测评机构进行测试，测试类型多为验收测试和确认测试。而这类软件测试项目的流程主要包括：分析测试需求、制定测试计划、测试设计、实施测试、建立和更新测试文档。软件测试项目的质量保证则是针对软件测试的过程和产品开展质量保证活动。一般建议每个软件测试项目组分配一名项目级的质量保证工程（也成为 QA 工程师），从事质量保证活动。项目的质量保证工程师向项目负责人和技术总监负责。下图为第三方软件测评机构软件测试项目的质量保证过程概要图。

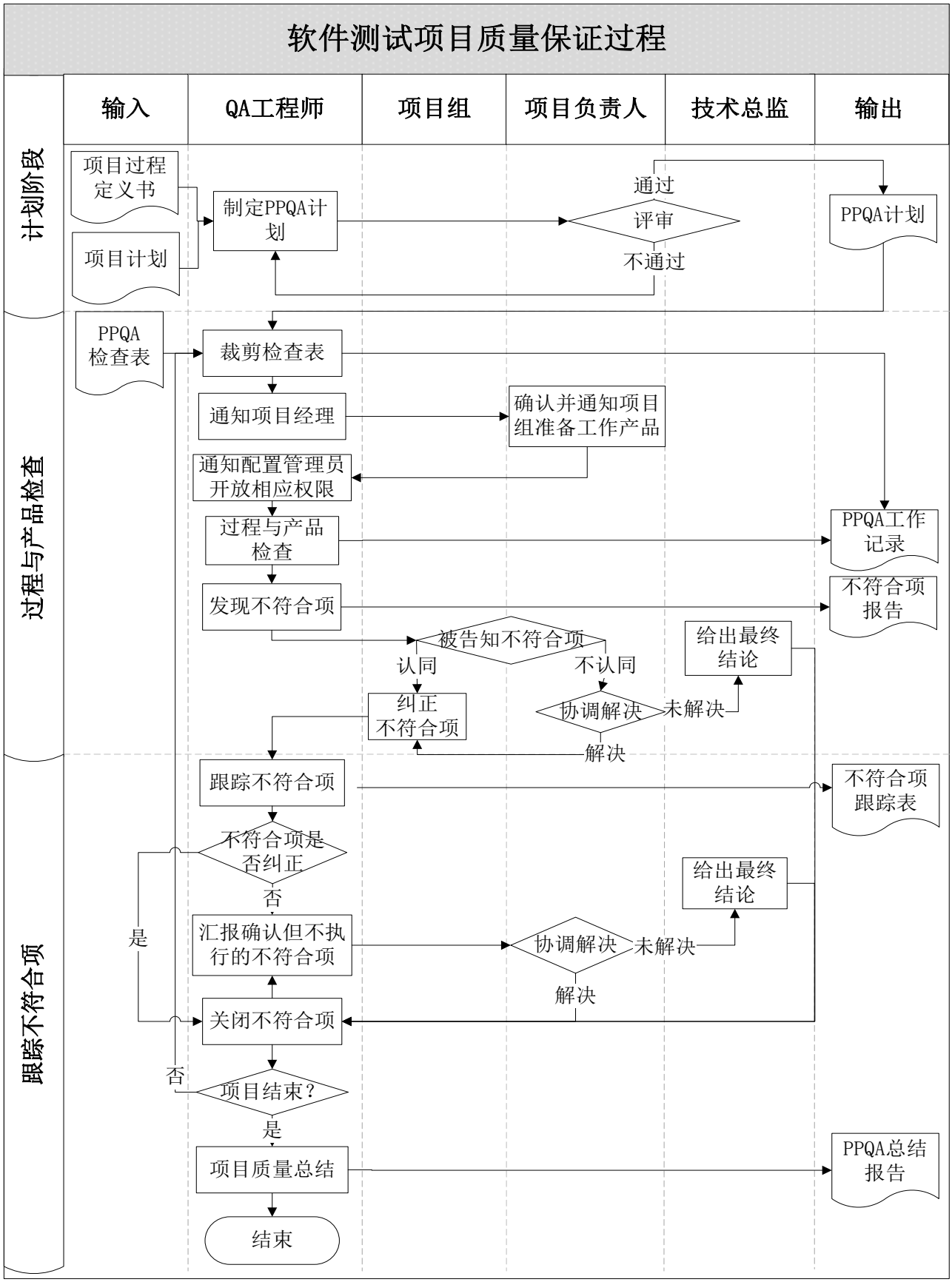


图 1 软件测试项目质量保证过程概要图

四、软件测试项目项目质量保证过程

1. 软件测试项目项目质量保证活动周期

软件测试项目项目质量保证贯穿于测试活动的始终，一般软件测试项目的质量保证活动自《测试方案》已经评审通过后开始启动，至《项目质量保证计划》执行完毕、所有不符合项被关闭/认可结束。

2. 制定质量保证计划

（1）裁剪《项目级项目质量保证检查表》

软件测评机构根据自身业务特点制定组织级别的《项目级项目质量保证检查表》，不同的测试项目可以根据自身需要进行裁剪。

在项目项目质量保证活动之初，项目级质量保证工程师根据《测试方案》剪裁该项目的《项目级项目质量保证检查表》。

（2）制定《项目质量保证计划》

项目级质量保证工程师根据单位的测试过程控制程序、已经评审通过的《测试方案》和裁剪后的《项目级项目质量保证检查表》，确定需要检查的主要过程及相应的过程产品，并估计检查时间和人员，制定《项目质量保证计划》。

项目级的日常检查活动按照项目开始后的自然周进行，跟踪检查活动根据不符合项的计划解决时间进行检查。

项目级质量保证工程师根据项目规则设定配置审计活动的频率，发生配置变更、基线变更和基线发布时追加检查。

（3）评审《项目质量保证计划》

项目级质量保证工程师组织《项目质量保证计划》评审，经由项目负责人审查，并由技术总监审核通过，输出《项目质量保证计划》。

没有通过评审的《项目质量保证计划》，项目级 QA 工程继续修改，直至项目负责人、技术总监分别审核通过。

3. 过程与产品检查

（1）实施审计

项目级质量保证工程师和项目负责人确定本次审计的时间、地点、参加人员等。

项目级质量保证工程师通知配置管理员开放某些权限。

项目负责人通知项目组成员准备相关工作产品。

项目级质量保证工程师根据《项目质量保证计划》、《项目级项目质量保证检查表》，在配置库中检查项目实际执行过程和相关工作产品是否符合既定的规范。

（2）记录结果

项目级质量保证工程师在《项目质量保证工作记录》中如实记录本次质量检查结果，并将不符合项记录在《不符合项跟踪表》中。

（3）协商纠正措施

项目级质量保证工程师应当与项目负责人分析不符合项原因、协商改进措施，并通知相关人员负责改进措施的实施。

对于项目组相关人员不认同的不符合项，项目级质量保证工程师将该不符合项汇报给项目负责人，由项目负责人协调解决。

如果还未能解决，项目级质量保证工程师将该不符合项汇报给技术总监，并由技术总监给出最终结论。

4. 跟踪不符合项

（1）跟踪不符合

项目级质量保证工程师根据《不符合项跟踪表》跟踪不符合项的解决过程，记录不符合项的状态，直到不符合项被解决/认可为止，统计数据、总结经验教训，并呈报给项目负责人和技术总监。

（2）疑难问题解决

对于问题相关人员承诺但不执行的不符合项，项目级质量保证工程师将该不符合项汇报给项目负责人，由项目负责人协调解决。如果还未能解决，项目级质量保证工程师将该不符合项追踪情况汇报给技术总监，并由技术总监给出最终结论。

（3）总结报告

每次检查活动跟踪完成后项目级质量保证工程师应及时将本次质量检查的结果、经

验教训通报给所有相关的人员。

项目级质量保证工程师定期分析机构内共性的质量问题，给出质量改进措施，以便于测评机构管理体系改进。

项目结束后，项目级质量保证工程师分析不符合项的发生，编写《项目质量保证总结报告》。

五、相关记录

（1）《测试方案》

（2）《项目级项目质量保证检查表》

（3）《项目质量保证计划》

（4）《项目质量保证工作记录》

（5）《不符合项跟踪表》

（6）《项目质量保证总结报告》