目 录

测试女巫-自动化实践篇

作者：妞妞

将统计学_6 sigma 应用到测试技术中，需要使用的 6 sigma 工具如下:
QFD，DFMEA ，量测系统检验，Johnson Transformation

一、前言:

测试女巫又来啦，大家还记的吗？前两次我们主要以“找到 bug 产生的原因”以及“提高 bug 产出效率”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 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。

那这次我们介绍什么呢？大家还记得第三十二期和第三十三期妞妞发表的两篇文章吗？“嵌入式产品自动化测试”和“路由器自动化测试”，对的在我们的测试中引进自动化测试是可以提高工作效率，因为很多机械的工作可以交给自动化工具完成，测试工程师可以做更有价值的工作……听起来非常美好啊~~但是，真的只要引进自动化就可以把大幅度提高工作效率吗？在我们引进自动化测试工具引进我们工作中这两年的时间，我们就发现现实的执行阶段并不是这样美好的~

OK，我们来列举几件:

1. 你的团队选哪些原本执行手动测试的工程师来开发自动化测试工具，是不是存在有的工程师花了很长时间开发的脚本，却因为开发脚本品质问题，需要重新返工，耗时耗人力，反而最终测试的效率反而不如直接执行手动测试?

2. 待测物所有的功能或者所有的测项都适合开发自动化脚本吗?是不是存在因为功能太复杂，开发以及维护所花费的人力以及精力大大超过你的预期?

3. 所有的软件版本都适合自动化脚本测试吗？是不是一拿到软件版本我们就可以启动我们的自动化脚本测试了，是不是不同品质的软件，都可以执行自动化测试?

我们在实际执行自动化测试时，是遇到很多困难，但是请不要轻易说：自动化测试只是“看上去很美”，我们不是有另一个“尚方宝剑”：6 sigma 吗，能不能使用 6 sigma 的理论以及

方法来分析我们自动化测试，找出自动化测试在开发阶段和执行阶段影响工作效率的因子，并找出高效引进自动化测试的方法？

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：QFD，DFMEA，量测系统，Johnson Transformation。在第三十四期的杂志中已经介绍了柏拉图，主效应图，交互作用图的原理和用法，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

1、QFD

它的全称是：Quality Function Deployment（质量功能展开）

它的作用是把顾客对产品的需求进行多层次的分析，转化为产品的设计要求、零部件特性、生产要求的质量工具，用来指导产品的健壮设计和质量保证。它主要是用在概念设计阶段，但是在优化设计和验证阶段也可以发挥辅助作用。

它主要功效如下:

1) 从顾客的角度对市场上同类产品进行评估

2) 从技术的角度对市场上同类产品进行评估

3) 确定顾客需求和技术需求的关系及相关程序

4) 分析并确定各技术需求相互之间制约关系

5) 确定各技术需求的目标值

2、DFMEA

它的全称为:Design Failure Mode and Effects Analysis ，它实际上是 FMA(故障模式分析)和 FEA（故障影响分析）的组合。它对各种可能的风险进行评价分析，以便在现有技术的基础上消除这些风险或将这些风险减小到可接受的水平，它是“事前的行为”而不是“事后的行为”，它是在产品设计前进行。

3、测量系统（MSA）

全称为 Measurement Systems Analysis

任何与测量相关的事物组合在一起就构成了所谓的测量系统，

测量的结果存在误差，也可以理解为测试结果是 Fail 的，那么问题来了：结果的 Fail 是待测物本身的差异引起的误差还是测量系统产生的差异引起的误差。以公式来表明这种关系：

总误差=待测物本身的差异引起误差+测量系统引起的误差

如果测量系统引起的误差超过了待测物本身的差异引起的误差，则此测试结果就是不可信的，所以在执行测试前，需要分析“测量系统引起的误差”是否在可接受范围内。

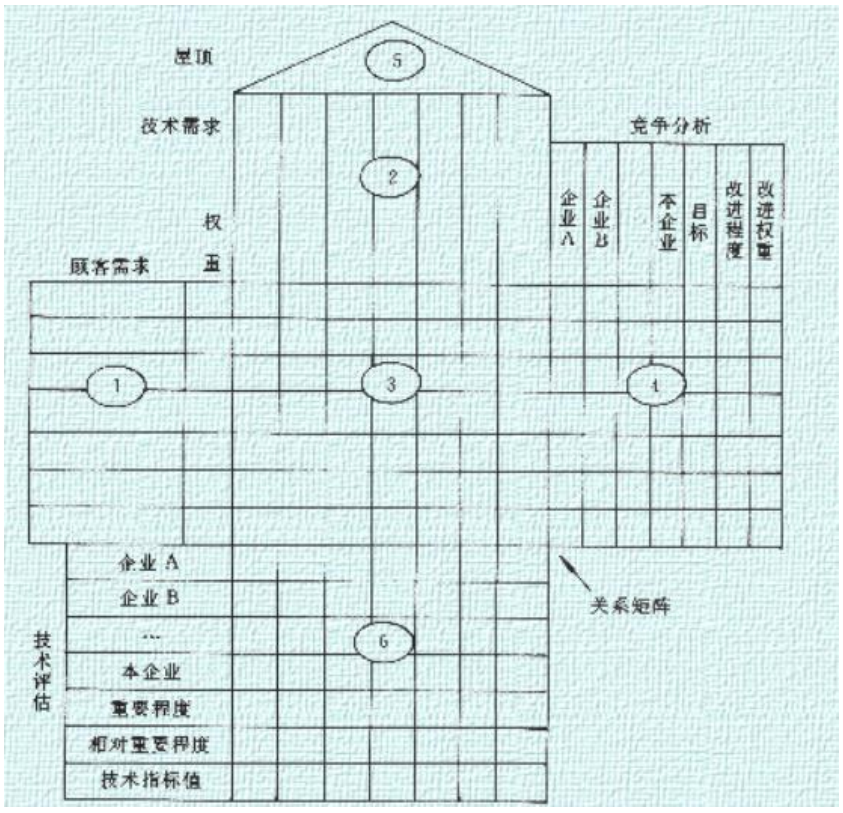
4、Johnson Transformation

很多 6 sigma 工具都要求分析的数据为正态分布，如果需要分析的数据不是正态分布，可以先使用 Johnson Transformation 此工具转换为正态分布

三、6 Sigma 常用工具介绍

1、QFD

质量屋的说明，如【图 1】



【图 1】

1) 【图 1】中标号为 1 的部分为各项顾客需求，并这些需求按照性能、可信度、安全性、适应性、经济型等进行分类，并对这些需求按照相互间的重要度进行标注，具体采用 1-10 数字

分 10 个级别标定各个需求的重要度。数值越大，重要度越大，反之，重要度越低。

2) 【图 1】中标号为 2 的部分为各项技术需求

这些技术需求必须符合以下要求:

a) 针对性，即技术需求要针对所配置的顾客需求

b) 可测量性，为了便于实施对技术需求的控制，技术需求应该是可测量的。例如如果顾客要求“希望产品的使用寿命长”时，对应的技术需求配置为“使用寿命”

3) 【图 1】中标号为 3 的部分为顾客需求与技术需求之间的关系

采用 1-10 数字分 10 个级别标注技术需求对各个顾客需求的贡献和影响程度，数值越大，重要度越大，反之，重要度越低。

4) 【图 1】中标号为 4 的部分为同级别不同竞争公司之间的关系

对各项顾客需求与其它竞争对手的对比分析，此部分在我们下面的实例中并没有用到。

5) 【图 1】中标号为 5 的部分为各项技术需求之间的关系

用于分析各项技术需求之间是否存在相互制约，设置相互矛盾的关系

6) 【图 1】中标号为 6 的部分为最终各项技术需求的排名以及与竞争公司的排名

2、DFMEA

DFMEA 表格说明

设计潜在的失效模式及后果分析（DFMEA）

_____系统_____子系统											DFMEA 编号：_____						
_____部件_____ 设计责任部门：_____											页码：第 页码 共 页						
车型年/车辆车型：_____ 关 键 日 期：_____											编制：_____						
主 要 参 加 人：_____											DFMEA 日期(编制)：_____(修订)_____						
													措施结果				
项目功能要求	潜在的失效模式	潜在的失效后果	严重程度S	级别	潜在失效原因/机理	频度O	现行设计控制预防	现行设计控制探测	探测度D	风险顺序RPN	建议措施	责任目标完成日期	采取的措施	严重程度S	频度O	探测度D	风险顺序RPN

【图 2】

1)项目功能要求

填入所分析的项目或者接口的功能，用尽可能简明的文字来说明被分析的项目要满足设计的功能。

2)潜在的故障模式

每项功能对应一种或者几种的故障模式，要尽量列出破坏此功能的所有模式。

3)潜在故障后果

每项故障模式都会有相应的故障后果，要尽量列出故障的最终影响，即最严重的影响

4)严重程度

对一个假定失效模式的最严重影响的评价等级，等级为 10-1；10 的严重度最高，数字越小，严重度越低具体的定义以一个汽车的 DFMEA 严重度定义如【图 3】，注意不同的产品严重度的评价标准会有所不同。

DFMEA 严重度评价标准（LQ）

后果	判定准则：后果的严重度	严重度数
无警告的严重危害	这是一种非常严重的失效形态，它是在没有任何失效预兆的情况下影响到行车安全或不符合政府法规。	10
有警告的严重危害	这是一种非常严重的失效形态，它是在具有失效预兆前提下所发生，影响到行车安全或不符合政府法规。	9
很高	车辆/系统无法运行，丧失基本功能，顾客极度不满。	8
高	车辆/系统能运行，但性能下降，顾客非常不满意。	7
中等	车辆/系统能运行，但舒适性或方便性项目失效，顾客不满意。	6
低	车辆/系统能运行，但有些舒适性或方便性项目性能下降，顾客有些不满意。	5
很低	装配和外观/尖响和卡嗒响等项目不舒服，多数（75%）顾客能感觉到有缺陷。	4
轻微	装配和外观/尖响和卡嗒响等项目不舒服，有一半（50%）顾客能感觉到有缺陷。	3
很轻微	装配和外观/尖响和卡塔响等项目不舒服，有辨别能力的顾客（25%以下）顾客能感觉到有缺陷。	2
无	无可辨别的后果。	1

【图 3】

5)潜在失效原因/机理

对于故障，即每个失效模式都要尽可能的列出此故障可能的原因

6)频度

指的是某一故障的原因或机理出现的可能性，等级为 10-1；10 的严重度最高，数字越小，严重度越低，具体的定义如【图 4】

失效可能性	可能的失效比率	分数
非常高：失效几乎不可避免	>= 1/10	10
	1/20	9
高：重复发生过的失效	1/50	8
	1/100	7
中等：偶尔失效	1/500	6
	1/2000	5
	1/10000	4
低：极少失效	1/100000	3
	<=1/1000000	2
极低：不太可能失效	防呆措施控制	1

【图 4】

7)现行设计控制预防

根据故障的潜在起因可确定预防的措施

8)探测度

是否可以通过现有的机制探测到这种失效模式，等级为 10-1；10 的可探测度最低，数字越小，可探测度越高

9)风险顺序(RPN)

RPN=S*O*D

S:Severity(严重度)

O:Occurrence（频度）

D:Detection(探测度)

此参数是对设计风险性的度量，它的取值在 1-1000 之间，数值越高，风险越高，对于 RPN 很高，且严重度很高的故障，一定要高优先级进行处理

10)建议措施

当失效模式按 RPN 派出顺序后，就应该首先对那些 RPN 值最高，且严重度最高的参数采取纠正措施，这些措施都是为了减少频度、严重度以及探测度的值。一般情况下严重度不会发生变化，但是可以通过修改设计来实现

11)措施结果

采取措施后的风险顺序值的变化，包括严重度，频度以及探测度

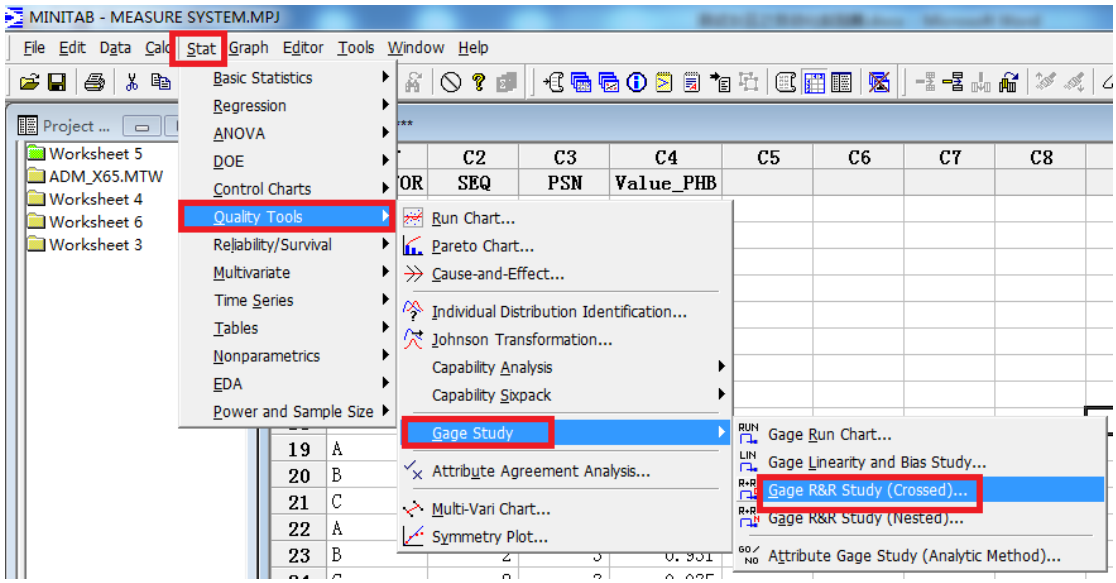
3、量测系统

1)首先在 Minitab 中建立一个表格，其中 Operator 指的是不同的操作者，SEQ 指的是执行的顺序，PNS 指的是待测物有几件，Value 指的是使用此量测系统测试的结果。

C1-T	C2	C3	C4
OPERATOR	SEQ	PSN	Value
A	1	2	0.956
B	1	2	0.917
C	1	2	0.965
A	2	2	0.980
B	2	2	0.956
C	2	2	0.955
A	3	2	0.945
B	3	2	0.959
C	3	2	0.960
A	1	3	0.940
B	1	3	0.931
C	1	3	0.936
A	2	3	0.932
B	2	3	0.931
C	2	3	0.935
A	3	3	0.934
B	3	3	0.939
C	3	3	0.932
A	1	4	0.913
B	1	4	0.917
C	1	4	0.916

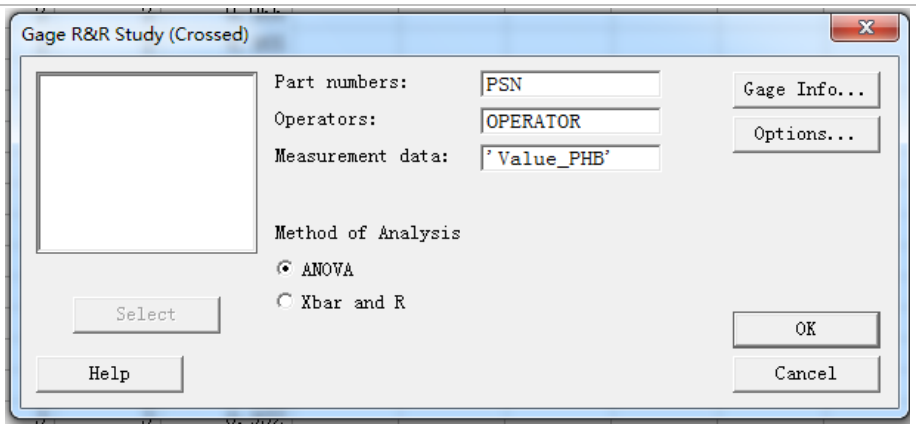
【图 5】

2)在 Minitab 中选择 QualityTools->GageStudy->GageR&RStudy(Crossed)，如【图 6】



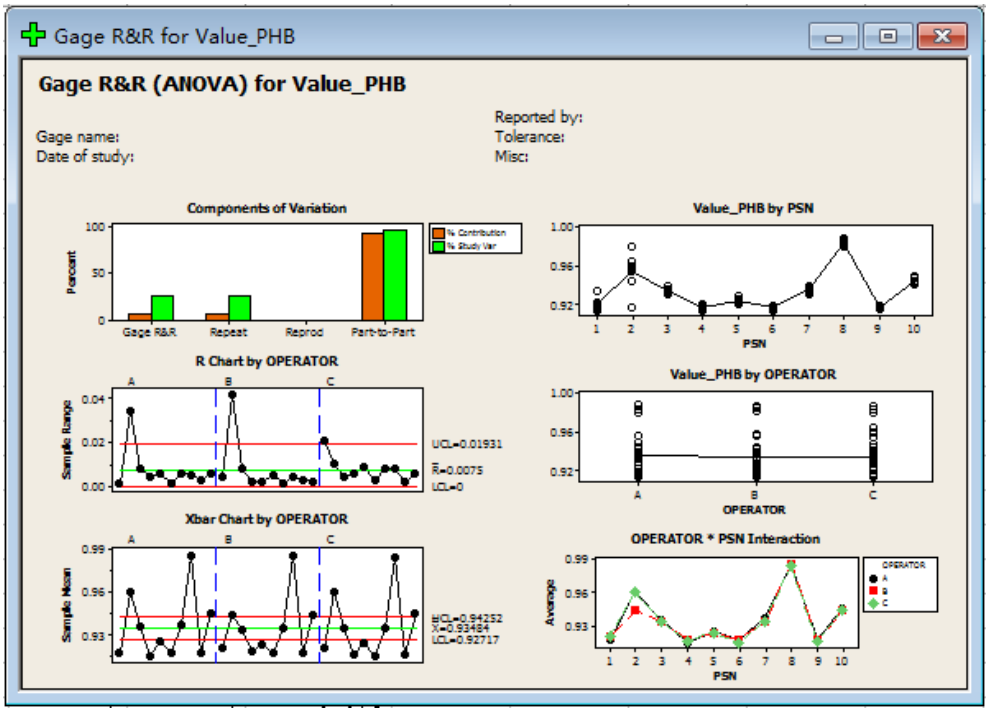
【图 6】

3)选择 GageR&RStudy(Crossed)后，选择 PartNumber 选择代表待测物有几件的栏位，



【图 7】

4)点击 OK 后会出现如【图 8】



【图 8】

5)测试结果的文字信息如【图 9】

根据文字结论进行进一步分析:

a.贡献度: %Contribution=3.4%

b.%StudyVar:18.47%

c.NDC:7

量测系统可接受的标准如下，如只能达到或者无法达到“需改善”的标准，则说明此量测

系统需要改善，不能直接使用

	GRR(%Contribution)	GRR(%SV)	NDC
OK	< 2%	< 10%	> 10
可接受	2 – 7.7%	10-30%	5-10
需改善	> 7.7%	> 30%	< 5

Source	DF	SS	MS	F	P
PSN	9	0.0400671	0.0044519	655.525	0.000
OPERATOR	2	0.0000024	0.0000012	0.178	0.838
PSN * OPERATOR	18	0.0001222	0.0000068	0.330	0.994
Repeatability	60	0.0012347	0.0000206		
Total	89	0.0414265			

Two-Way ANOVA Table Without Interaction

Source	DF	SS	MS	F	P
PSN	9	0.0400671	0.0044519	255.911	0.000
OPERATOR	2	0.0000024	0.0000012	0.070	0.933
Repeatability	78	0.0013569	0.0000174		
Total	89	0.0414265			

Gage R&R

Source	VarComp	%Contribution (of VarComp)
Total Gage R&R	0.0000174	3.41
Repeatability	0.0000174	3.41
Reproducibility	0.0000000	0.00
OPERATOR	0.0000000	0.00
Part-To-Part	0.0004927	96.59
Total Variation	0.0005101	100.00

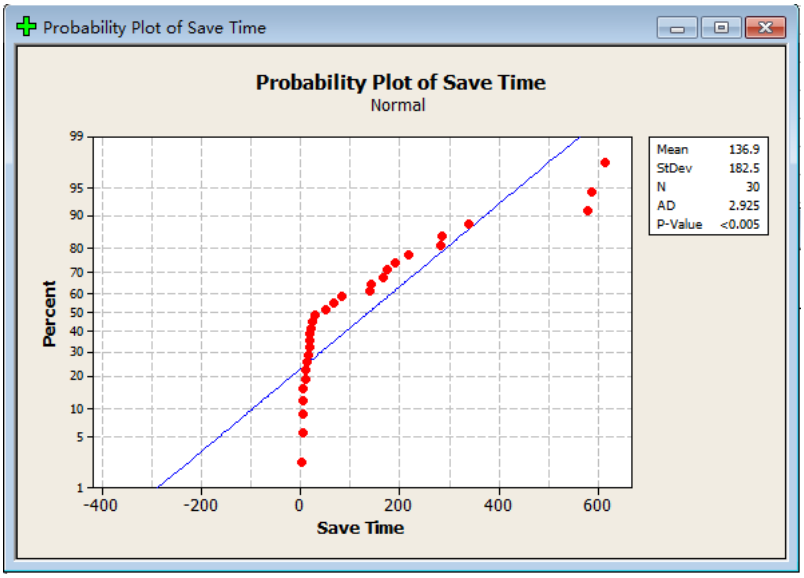
Source	StdDev (SD)	Study Var (6 * SD)	%Study Var (%SV)	%Tolerance (SV/Toler)
Total Gage R&R	0.0041709	0.025025	18.47	5.01
Repeatability	0.0041709	0.025025	18.47	5.01
Reproducibility	0.0000000	0.000000	0.00	0.00
OPERATOR	0.0000000	0.000000	0.00	0.00
Part-To-Part	0.0221974	0.133184	98.28	26.64
Total Variation	0.0225858	0.135515	100.00	27.10

Number of Distinct Categories = 7

【图 9】

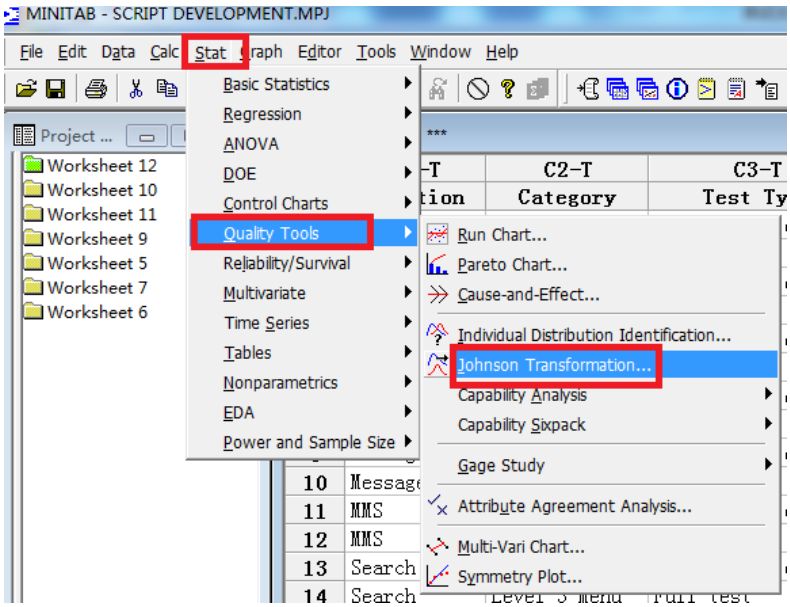
4、JohnsonTransformation

对于我们希望研究的数据，首先分析它是否为正态分布（此部分内容已经在上两期的杂志中有详细的说明，这里就不再赘述）如【图 10】



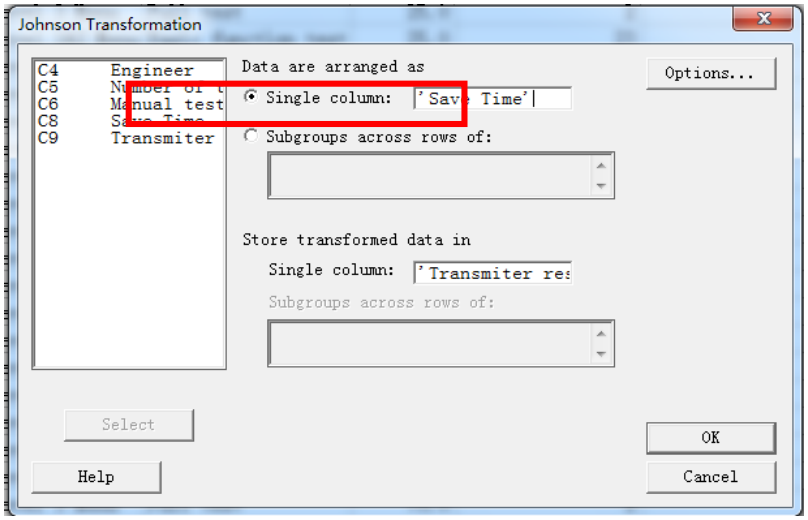
【图 10】

因为不是正态分布，很多 Minitab 的工具都不能使用，例如 ANOVA,Regression 等等，所以我们需要先对这些数据进行转换的动作，进入 QualityTools->JohnsonTransformation 如【图 11】,



【图 11】

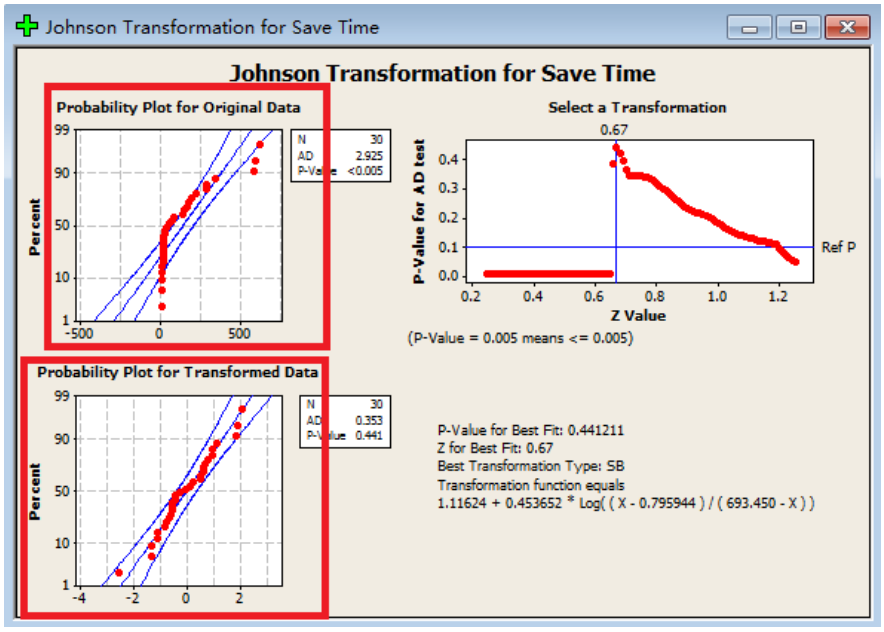
在 Singlecolumn 中选择需要转换的数据【图 12】



【图 12】

4)点击 Ok 后出现如【图 13】,

第一个红框是未转换的数据，第二个红框是转换后的数据



【图 13】

四、应用到实际工作中_自动化测试实践篇

1、列出目前存在的矛盾

1)软件工程师的要求:

a.能否有更多的时间 supportRD 解决 bug,例如抓 Log，追踪 bug?

b.应该更快的响应测试需求，尤其是计划外的需求

c.能否既能满足测试的全面性，又能将所有隐藏较深的 bug 尽早发现出来

2)项目经理的要求

a.需要这样多的测试时间吗？不能缩短测试时间吗

b.需要这样多的人力吗？不能减少人力吗

3)我们测试部门的要求

a.不同项目的测试需求能不能不要太集中，能否分散平均点？

b.能否给足够的时间测试，这样才能同时满足测试的全面性和测试的深度

c.每个项目能否配置足够的人力，这样才能保证测试的质量使用 QFD

找出解决目前存在矛盾的解决方法，如【图 14】

从【图 14】的结论可以看出解决他们的要求，最重要的解决方法是：执行手动测试和执行自动化测试

Planning QFD

	编写测试计划									
	合理配置人力		◎							
	缩短测试时间			○						
	提高发现bug的效率			◎	◎					
	执行手动测试				◎					
	使用自动化进行所有功能测试			◎	◎	○				
	编写效率高的测试用例				○	◎	○	○		
	Functional Performance Requirements / Substitute Quality Characteristics or Metrics									
	Customer Requirements	WEIGHT	编写测试计划	合理配置人力	缩短测试时间	提高发现bug的效率	执行手动测试	使用自动化进行所有功能测试	编写效率高的测试用例	Comments
合作部门的要求	及时开始进行测试任务	8		5			3	10		
	测试时间尽量短，同时要保证测试bug的品质	10		5	6	5	4	9	7	
	保证测试全面性	9	5	4			5	9		
	有足够的时间协助SW RD解决bug	6			6	7	2	9	4	
	保证测试深度	8		6		6	10	6	8	
客户需求	对于迭代版本需要定期进行基本功能的测试，以保证其基本品质	8	6				2	9		
测试部门的要求	测试时间充足	8	6				2	4		
	不同项目的测试任务分布均匀，不要所有的任务集中在一起	6	9				2	8		
	最基本功能的测试尽量减少	5						9		
	非计划内的测试尽量减少	6	9					8		
	测试FW应该保证基本的品质	8	4	5	7			8		
公司的要求	减少测试的成本与人力	10		5	7	8	2	8	5	
	Company Scaled Importance: GAP / Risk	Rank	6	3	7	4	2	1	5	

【图 14】

3、使用 DFMEA 找出解决方案存在的问题

如【图 15】

項目/功能 (Component/ Function Requirements)	要求(Requirements)	潛在失效模式 (Potential Failure Mode)	失效的潛在效果 (Potential Effects of Failure)	嚴重性 S	等級 C	失效的潛在原因與結構 (Potential Causes /Mechanism of Failure)	發生性 O	現行設計管制 (Current Design Control)		偵測性 D	S O D	R P N
								預防(Prevention)	偵測(Detection)			
Cable												
使用自动化脚本测试基本功能	能够完成功能的测试	脚本不稳定,待测版本不稳定,开发和维护时间过长	无法使用自动化脚本实现基本功能的的测试	7	◇	开发脚本质量不高待测软件版本质量较差，无法正常执行脚本，软件Fw的界面经常发生变化	7	NA	实施基本功能自动化测试	6	776	294
使用手动测试进行所有功能的测试	在规定的时间能可以完成所有功能的测试	测试时间不够，人力不够	无法按时完成测试任务	7	◇	临时任务插单过多，需要执行的测试用例过多	9	NA	执行测试	6	796	378

【图 15】

1)执行自动化测试

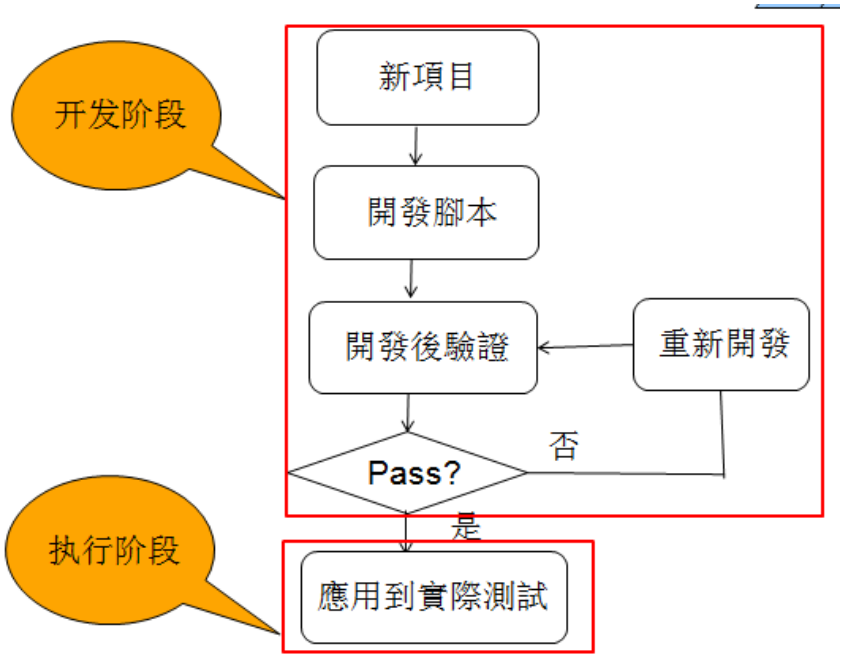
可以保证测试全面性，对于测试及时性要求以及不均的测试任务都

可以满足，利于减少人力成本，但测试的度比较差。

2)执行手动测试

可以加强测试的深度，但是测试的全面性比较差，且不利于减少人力成本。

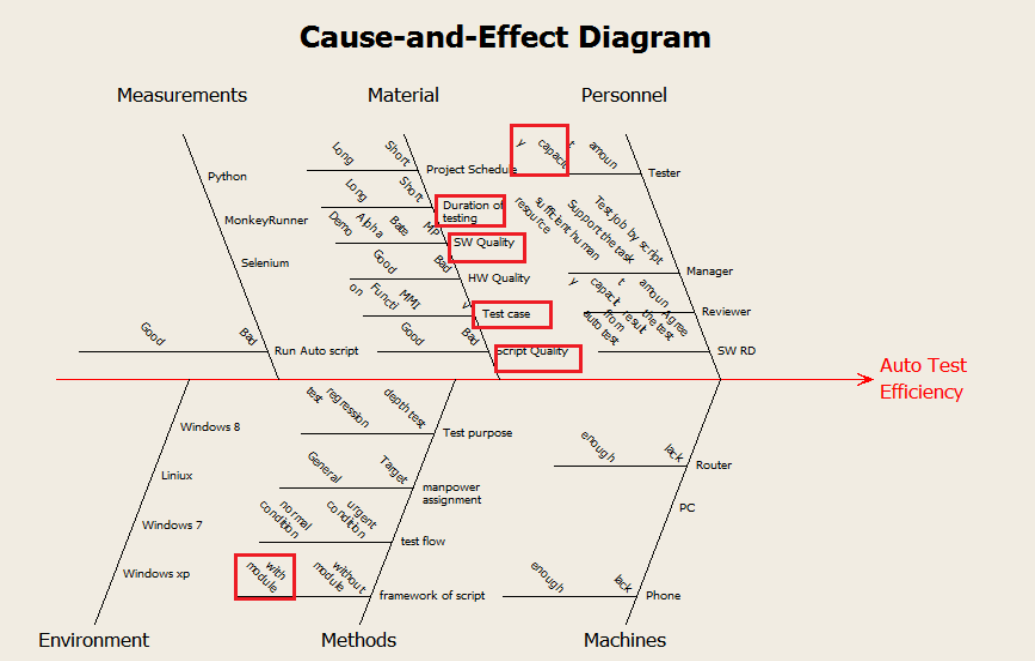
4、目前的流程



【图 16】

5、使用鱼骨图分析原因

如【图 17】



【图 17】

根据鱼骨图的分析，提炼出关键因子如【图 18】

自动化相关阶段	原因	说明
与执行阶段相关	DurationofTesting	如果项目的测试时间持续越长，则节省的人力越多
	AutotestscriptQuality	不同质量的脚本，会影响脚本执行的效率
	SWQuality	软件的质量分为 DemoAlphaBetaMP 此 4 大阶段，不同阶段的软件质量有比较大的差异
与开发阶段相关	Tester:Capacity	不同能力的测试人员，开发脚本所消耗的时间和脚本的质量会有很大的差异
	Testcase	什么样的 Testcase 适合开发成为自动化脚本
	WithModule	脚本的开发，如果实现模块化开发，则效率会大大提高

【图 18】

6、量测系统分析

1)问题描述:

在我们原有手动测试流程的基础上，增加了自动化测试系统，为了证明我们开发的自动化系统符合要求，所以需要对我们的自动化测试系统进行量测。

2)量测系统说明

挑选测试人员 3 名，针对某一产品 10pcs，重复随机针对 Settings 此功能测试 3 次,得到如下数据（因为 Settings 是最复杂的一个功能，它汇集了手机各个功能的设置项目）如【图 19】

C1-T	C2	C3	C4
OPERATOR	SEQ	PSN	VALUE
A	1	1	1.217
B	1	1	1.223
C	1	1	1.234
A	2	1	1.218
B	2	1	1.219
C	2	1	1.216
A	3	1	1.217
B	3	1	1.220
C	3	1	1.213
A	1	2	1.253
B	1	2	1.249
C	1	2	1.265
A	2	2	1.276
B	2	2	1.256
C	2	2	1.255
A	3	2	1.245
B	3	2	1.259
C	3	2	1.260
A	1	3	1.237
B	1	3	1.231
C	1	3	1.236
A	2	3	1.231
B	2	3	1.231
C	2	3	1.235

【图 19】

3)分析结论如【图 20】

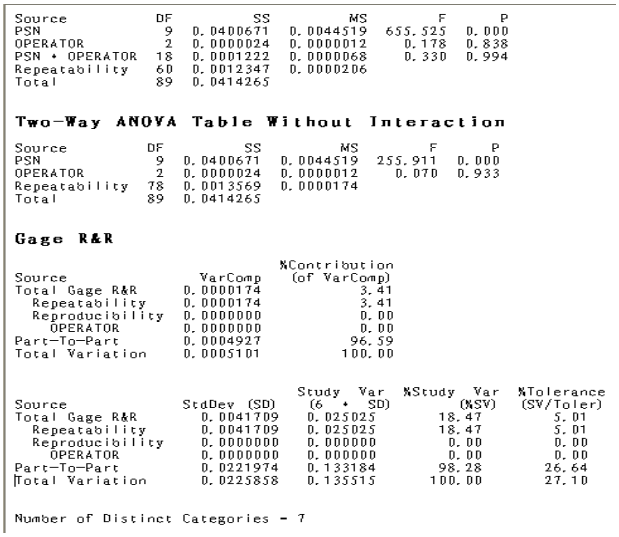
根据上述测试结果以及下表的判定标准，分析下面参数

按照上述结果进行分析，结果如下:

a.%Contribution=3.41%，按照规格是处于可以接受的范围

b.%StudyVar=0.0041709/0.0225858=18.47%，按照规格是处于可接受的范围

c.NDC=1.41*(0.0221974/0.0041709)=7,按照规格是处于可以接受的范围



```
Source            DF          SS          MS          F      P
PSN                9   0.04DD671   0.DD44519   655.525  0.DDD
OPERATOR           2   0.DDDDD24   0.DDDDD12     0.178  0.838
PSN * OPERATOR    18   0.DD01222   0.DDDD068     0.330  0.994
Repeatability     60   0.DD12347   0.DDDD206
Total             89   0.0414265

Two-Way ANOVA Table Without Interaction

Source            DF          SS          MS          F      P
PSN                9   0.04DD671   0.DD44519   255.911  0.DDD
OPERATOR           2   0.DDDDD24   0.DDDDD12     0.070  0.933
Repeatability     78   0.DD13569   0.DDDD174
Total             89   0.0414265

Gage R&R

                                 %Contribution
Source                VarComp     (of VarComp)
Total Gage R&R     0.DDDD174             3.41
  Repeatability    0.DDDD174             3.41
  Reproducibility  0.DDDDDDD             0.DD
    OPERATOR       0.DDDDDDD             0.DD
Part-To-Part       0.DD04927            96.59
Total Variation    0.DD05101           100.DD

                                  Study  Var  %Study  Var   %Tolerance
Source             StdDev (SD)    (6   *   SD)        (%SV)   (SV/Toler)
Total Gage R&R       0.DD41709   0.025025        18.47          5.D1
  Repeatability      0.DD41709   0.025025        18.47          5.D1
  Reproducibility    0.DDDDDDD   0.DDDDDD         0.DD          0.DD
    OPERATOR         0.DDDDDDD   0.DDDDDD         0.DD          0.DD
Part-To-Part         0.0221974   0.133184        98.28         26.64
Total Variation      0.0225858   0.135515       100.DD         27.10

Number of Distinct Categories = 7
```

【图 20】

判断标准如下图

	GRR(%Contribution)	GRR(%SV)	NDC
OK	< 2%	< 10%	> 10
可接受	2–7.7%	10-30%	5-10
需改善	> 7.7%	> 30%	< 5

综上，说明量测系统是可信赖的。

对于筛选出来的关键因子进行深入的研究

Input	Output	Control
DurationofTesting	测试的时间越长，节约的人力越可观	不可控
SWQuality	需要进一步研究，软件质量的差异对自动化效率有什么影响	可控
AutotestscriptQuality	需要进一步研究导致自动化脚本不能执行成功的原因是什么	可控
Tester:Capacity	什么类型的 Tester 比较适合	可控

	开发自动化脚本	
TestCaseandFunction	什么测项和功能适合开发自动化脚本	可控
WithModule	脚本的开发，如果实现模块化开发，则效率会大大提高	不可控

【图 21】

1)针对 SWQuality 搜集的资料如【图 22】:

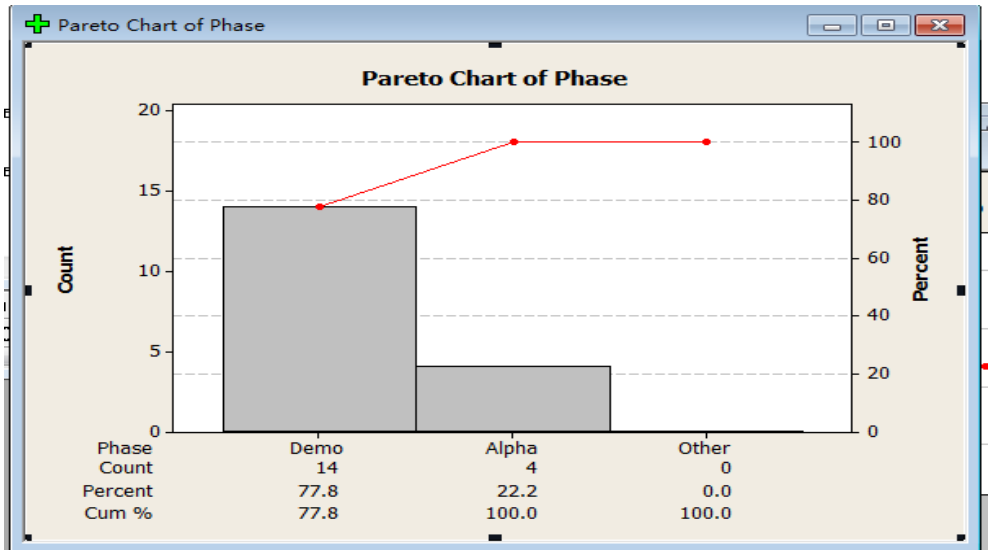
处于公司保密要求将项目名称设为不可见，即在软件不同阶段，不同项目在执行自动化测试时，重测次数

C1-T	C2-T	C3	C4
Phase	Project	Test Time	Retest Time
Demo	[redacted]	8	5
Alpha	[redacted]	12	3
Bata	[redacted]	8	0
MP	[redacted]	4	0
Demo	[redacted]	4	3
Alpha	[redacted]	4	0
Bata	[redacted]	2	0
MP	[redacted]	1	0
Demo	[redacted]	4	3
Alpha	[redacted]	5	1
Bata	[redacted]	4	0
MP	[redacted]	2	0
Demo	[redacted]	5	3
Alpha	[redacted]	4	0
Bata	[redacted]	3	0
MP	[redacted]	2	0

【图 22】

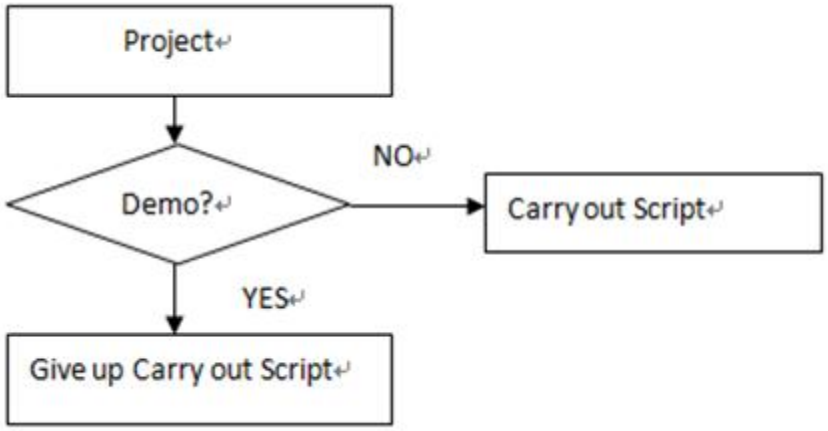
✓对于上述数据做柏拉图分析

如【图 23】可以看出在 Demo 阶段重测次数非常的多，比较浪费人力物力。



【图 23】

✓由此产生的流程如【图 24】



【图 24】

3)对 AutotestscriptQuality 分析

✓搜集的数据

即对于自动化脚本最容易产生的失败原因，针对不同的功能搜集的失败的次数如【图 25】

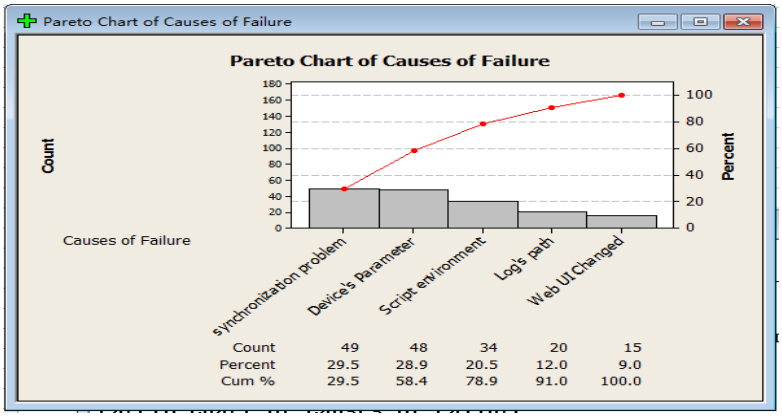
Causes of Failure	Function	Time
Log's path	Message	0
Device's Parameter	Message	0
Script environment	Message	1
Web UI Changed	Message	0
synchronization problem	Message	5
Log's path	E-mail	0
Device's Parameter	E-mail	8
Script environment	E-mail	6
Web UI Changed	E-mail	0
synchronization problem	E-mail	6
Log's path	Setting	0
Device's Parameter	Setting	2
Script environment	Setting	3
Web UI Changed	Setting	0
synchronization problem	Setting	5
Log's path	Phonebook	1
Device's Parameter	Phonebook	3
Script environment	Phonebook	4
Web UI Changed	Phonebook	0
synchronization problem	Phonebook	4
Log's path	Gallery	4
Device's Parameter	Gallery	6
Script environment	Gallery	3

【图 25】

✓通过柏拉图分析如【图 26】

从以下分析可以得出:

失效原因主要是: 同步的问题以及 Device 参数设置的失效概率接近 60%



【图 26】

✓针对两个最显著的失败原因下的对策

Parameter	解决方法
同步问题	我们是用 sleep(time)语句进行同步的，不同的 PC 和 Device 可能需要的时间会有差异，后续需要对此 time 留一些 buffer 防止因为 PC 的差异导致运行失败
Device 的设置	由于 Device 设置的参数不同，导致脚本运行的环境不一样可能会导致，后续需要细化脚本文档，在其中说明执行此脚本的前提条件

4)对 Tester:Capacity 分析

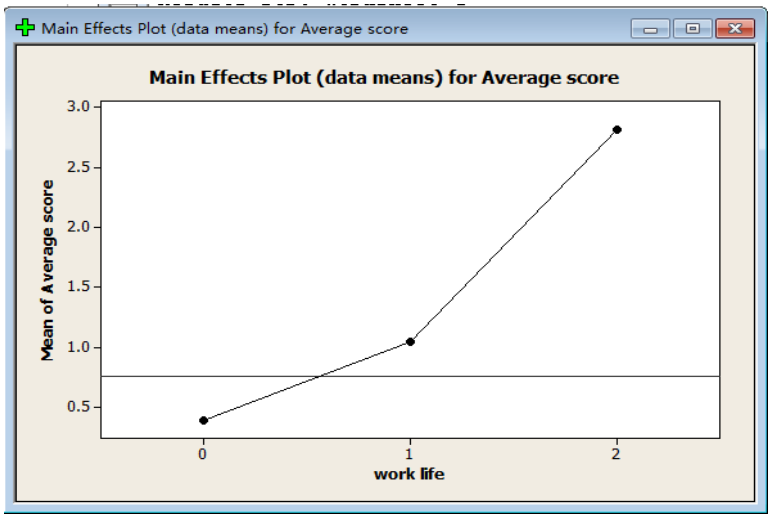
✓搜集资料如【图 27】

Member	Average score	work life
[illegible]	4.81	2.00
[illegible]	2.03	2.00
[illegible]	1.53	2.00
[illegible]	2.88	2.00
[illegible]	1.27	1.00
[illegible]	1.36	1.00
[illegible]	1.28	1.00
[illegible]	0.89	0.00
[illegible]	0.25	1.00
[illegible]	0.56	0.00
[illegible]	0.14	0.00
[illegible]	0.42	0.00
[illegible]	0.09	0.00
[illegible]	0.93	0.00
[illegible]	0.14	0.00
[illegible]	0.30	0.00
[illegible]	0.06	0.00
[illegible]	0.86	0.00
[illegible]	0.34	0.00

【图 27】

✓使用主效应图分析如【图 28】

从主效应图可以看出 Worklife 与 bugscore 有很大的关联性



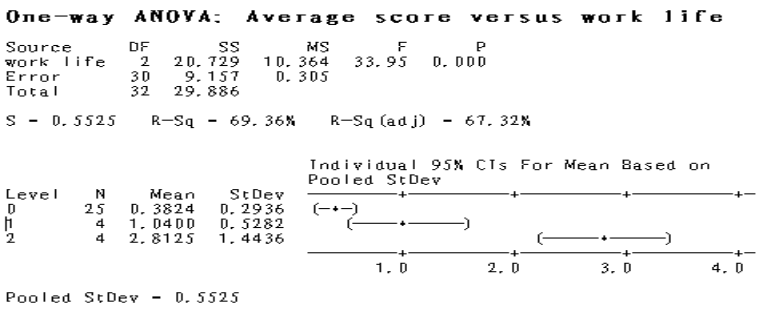
【图 28】

✓进一步分析具体有什么关联如【图 29】

Worklife:工作年限 1 年左右:bugscore 在 0->0.5 之间

工作年限 3 年左右的 bugscore 在 0.5->1.6 之间

工作年限 5 年左右的 bugscore 在 2.2->3.5 之间



【图 29】

✓搜集由【图 29】得出的结论，将人员分为 3 类:

A 类为工作年限在 5 年左右，bugscore 在 2.5->3.5 之间

B 类为工作年限在 3 年左右，bugscore 在 1->1.6 之间

C 类为工作年限在 1 年左右，bugscore 在 0.5 左右

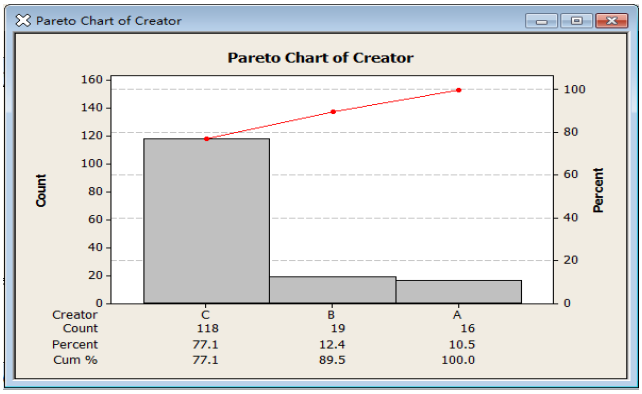
这些人员在不同项目相同 function 开发的自动化脚本，出错的次数

Tester: Capacity

Function	Time	Creator
E-mail	8	B
E-mail	6	B
E-mail	0	B
E-mail	6	B
Setting	0	A
Setting	2	A
Setting	3	A
Setting	0	A
Setting	5	A
Phonebook	1	B
Phonebook	3	B
Phonebook	4	B
Phonebook	0	B
Phonebook	4	B
Gallery	4	B
Gallery	6	B
Gallery	3	B
Gallery	6	B
Gallery	5	B
Calendar	4	C
Calendar	6	C

【图 30】

✓使用柏拉图分析如【图 31】



【图 31】

可以看出 C 类人员出错的几率非常之大

5)对于 TestCaseandFunction 进一步分析

✓数据搜集如【图 32】

表格的内容说明:

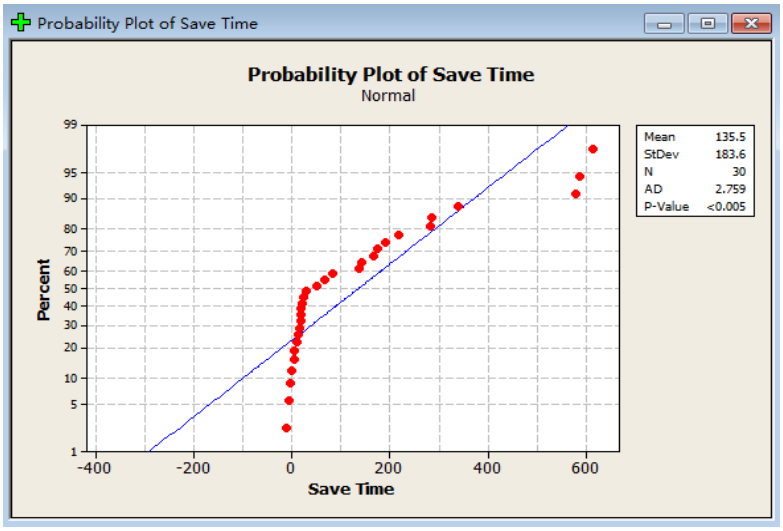
对于不同 Function 不同类型的 testcase 执行自动化测试节省的时间

Function	Category	Test Type	Save Time	Transmiter result
Browser	Level 1&2 Menu	Basic function test	283.0	0.94629
Browser	Level 3 Menu	Full test	23.0	-0.42966
Phonebook	Level 1&2 Menu	Basic function test	338.0	1.09233
Phonebook	Level 3 Menu	Full test	4.0	-1.32055
File manager	Level 1&2 Menu	Basic function test	190.0	0.67227
File manager	Level 3 Menu	Full test	1.0	-2.57177
Setting	Level 1&2 Menu	Basic function test	579.0	1.85106
Setting	Level 3 Menu	Full test	4.0	-1.32055
Message	Level 1&2 Menu	Basic function test	613.0	2.03689
Message	Level 3 Menu	Full test	51.0	-0.04021
MMS	Level 1&2 Menu	Basic function test	82.0	0.20037
MMS	Level 3 Menu	Full test	11.0	-0.79042
Search	Level 1&2 Menu	Basic function test	66.0	0.08910
Search	Level 3 Menu	Full test	6.0	-1.09920
Calendar	Level 1&2 Menu	Basic function test	284.0	0.94900
Calendar	Level 3 Menu	Full test	18.0	-0.54877
Caculator	Level 1&2 Menu	Basic function test	30.0	-0.30059
Caculator	Level 3 Menu	Full test	6.0	-1.09920
E-mail	Level 1&2 Menu	Basic function test	587.1	1.89067
E-mail	Level 3 Menu	Full test	10.0	-0.83788
Gallery	Level 1&2 Menu	Basic function test	217.0	0.75779
Gallery	Level 3 Menu	Full test	15.0	-0.63771
Music	Level 1&2 Menu	Basic function test	143.0	0.50223
Music	Level 3 Menu	Full test	21.0	-0.47384
Video	Level 1&2 Menu	Basic function test	138.0	0.48189

【图 32】

✓对于 savetime 进行正态分布检测如【图 33】

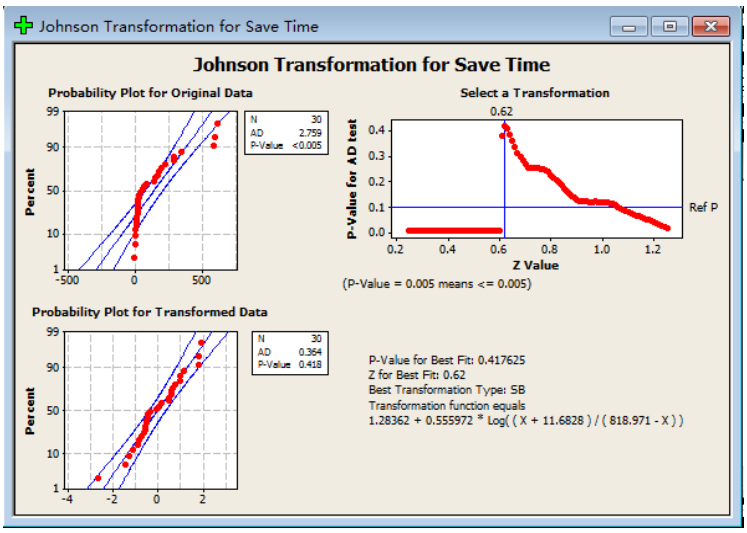
针对自动化测试不同类型的 Case 以及不同 Function 的节省时间进行分析对于 Savetime 进行正态检测，发现不是正态分布



【图 33】

✓对于此数据进行正态化的处理根据柏拉图的分析:

使用 JohnsonTransformation 将其转为正态分布，利于后续的分析



【图 34】

✓进行 ANOVA 分析如下图【图 35】

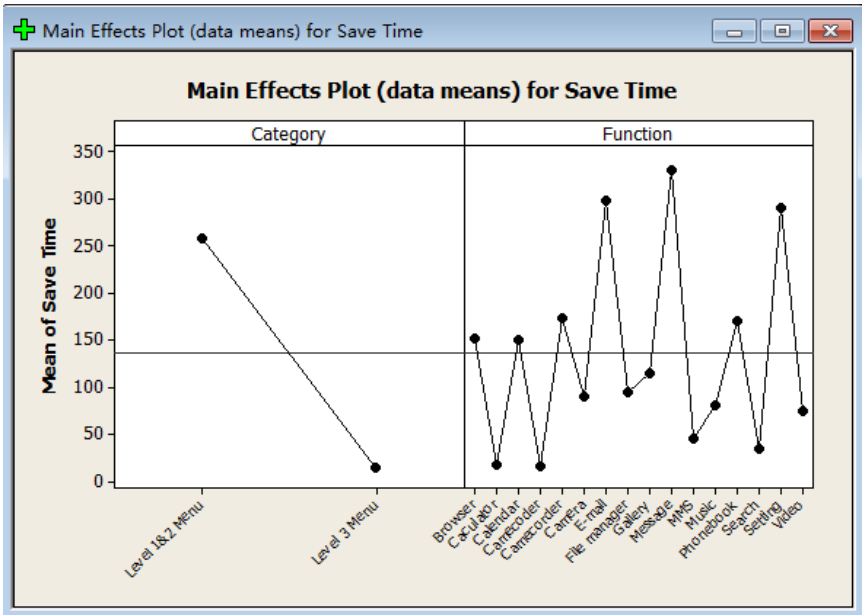
```
General Linear Model: Transmiter r versus Function, Category, Test Type

Factor     Type   Levels  Values
Function   fixed      16  Browser, Caculator, Calendar, Camecoder, Camecorder,
                          Camera, E-mail, File manager, Gallery, MMS, Message,
                          Music, Phonebook, Search, Setting, Video
Category   fixed       2  Level 1&2 Menu, Level 3 Menu
Test Type  fixed       2  Basic function test, Full test


Analysis of Variance for Transmiter result, using Adjusted SS for Tests

            Model
Source         DF  Reduced DF    Seq SS
Function       15          15    7.1248
Category        1           1   20.8213
Test Type       1          0+    0.0000
Error          12          13    4.6039
Total          29          29   32.5499

+ Rank deficiency due to empty cells, unbalanced nesting, collinearity, or an
     undeclared covariate. No storage of results or further analysis will be
     done.


S = 0.595100   R-Sq = 85.86%   R-Sq(adj) = 68.45%

Main Effects Plot (data means) for Save Time
```

【图 35】

✓进一步对 Testcase 和 Function 进行主效应图分析

如【图 36】Category 为 Level1&Level2Menu 对于节省时间是

显著因子 Function 并没有看出非常有规律的。



【图 36】

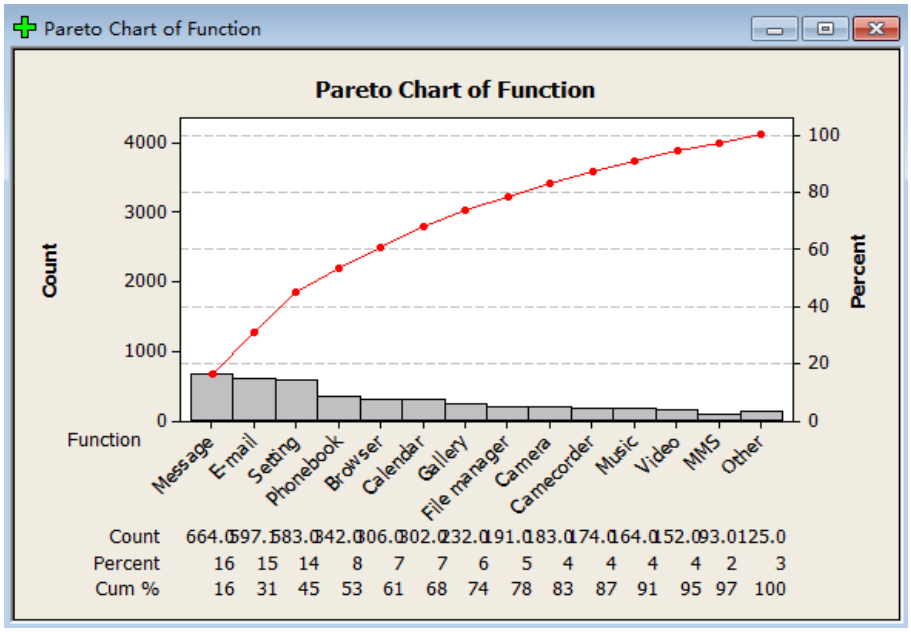
✓对于 Function 因子的进一步分析

使用柏拉图进行分析，如【图 37】

对于人力节省影响较大的功能为:

MessageE-mailSettingPhonebookBrowser

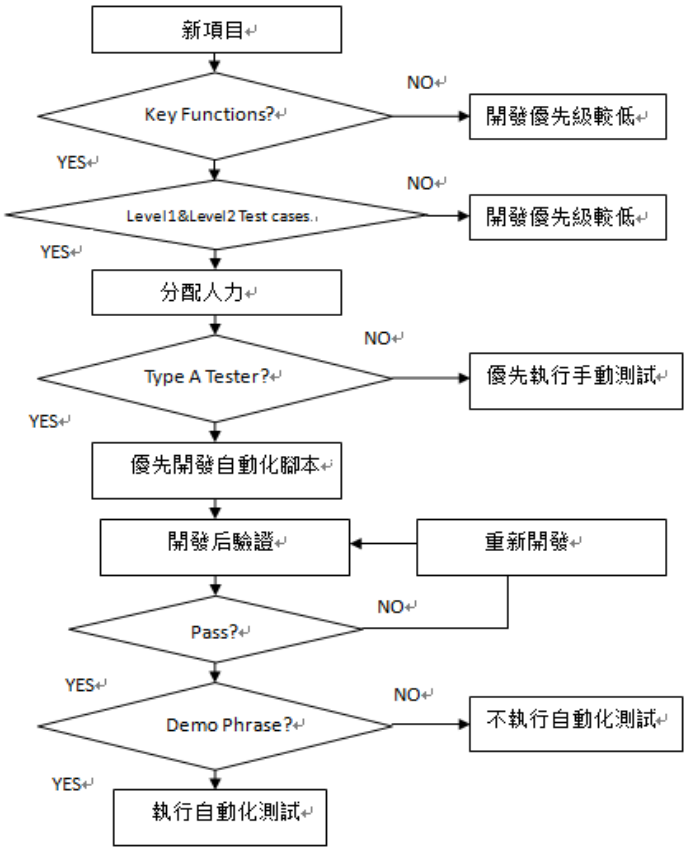
这些功能定义为关键功能



【图 37】

8、产生的最终流程

对于上面各个因子的分析产出一个最终的流程如【图 38】



【图 38】

五、总结

我们再一次给出了一个典型的例子来说明统计学如何应用到实际工作中，二十一世界最缺少的是什么：人才！我再加一句：“复合型人才！!”各位看官能否想象一下，你即能搭建自动化测试系统，又能玩转统计学即使用 6Sigma 的理论以及工具，且利用它们你可以找到如何高效率的开发自动化测试系统，以及如何能让自动化测试系统更好的发挥其作用进而减少人力。这不就是非常牛的“复合型人才吗”嗯！在下以为非常之给力，作为“失业即沦为贫民的无产阶级”的我而言这样的复合型才能，真的多多益善，只有让自己不断的内功修炼进而慢慢成为“多角度各方面”的“复合型人才”才能立于不败之地!!哇哈哈想想就觉得不管外面环境竞争有多激烈，只要我潜心修炼内功，就可以从容面对一切!!多么让人心旷神怡~~~

OK 冷静一下，我们来总结一下这次学的理论和工具：QFD 它实际上是一个“翻译工具”，经常开发项目的人应该有这样的体会，有效的沟通真的是一件非常重要的事情，如何将市场部人员或者客户的要求，转换为工程师看的懂的产品设计要求？它实际上体现出以客户的要求为导向，以客户的需求作为开发的重要指导思想。对于不会直面客户或者市场部人员的我们，可以转换一下思想，将与我们合作部门的需求当作客户的需求，这样我们就能更高效的与其它部

门进行沟通合作。DFMEA 它是一个“事前诸葛亮”，即在做一个新项目前，我们要根据之前的 bug 也可以理解为“失效模式”进行预测：做这些功能有可能会遇到的问题是什么，然后分析这些问题的风险顺序，按照风险顺序在问题被暴露出来之前下对策进行解决。这个工具给了我们一个非常好的工作流程，很多事情如果准备工作做得足，就会“事半功倍”！量测系统更是让我们思考，你的测试结果真的符合实际吗？是不是因为你的测试系统的异常造成测试结果异常；是不是因为不同的人员操作的差异导致的异常？这个工具让我们在测试前，就要把这些问题考虑进去，保证我们的测试结果是可信赖的。JohnsonTransformation 只是一个转换工具，对于待研究的数据不是正态分布的情况下，MiniTab 会有很多功能无法使用，只要使用此工具进行转换，就可以正常使用这些工具了。

对于 6sigma 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”！ ■

参考文献:

1、有关 DFMEA 基本概念介绍的网站：http://baike.baidu.com/link?url=4wQ2ENC-UFQTbaitzvUT4Q_1Drz35PWgRg7hjldxQyJfwsZfpXzVZ-R2HM7cR-2giFzgUYzeEasW7Egk8L6M4q

2、有关“量测系统”的介绍的网站：

http://doc.mbalib.com/view/b597a167b22af1a592287c4ab17fac32.html

3、有关 QFD 基本概念介绍的网站：

http://baike.baidu.com/link?url=_adQv_mLwCDA2j7PHN8NDsfoEimwS5HNeSEX2EQtmgpz_-uQ3KMw31okeQqwYFbUH6ptk-GvZvdcfxQuZS-lwa

作者简介

妞妞，某终端开发公司系统部测试课的课长，已经从事终端软件测试 10 年，从事软件测试团队管理工作 8 年，致力于研究自动化与手动测试结合进而产生高效率工作的方法。

◆ 作 者
潘 杰

国际化本地化测试中的基于 web 的 gui 自动化测试技术探讨思考

如何利用 qtp 进行 GUI 的国际化本地化测试

一、 简介

由于软件测试行业的分工越来越细致和明确，以及国际上的一些大型软件公司譬如 MS，IBM、HP 等企业，基于成本和市场考量还有 IT 业务本身的调整，会将在美日欧市场的一部分测试需求转移到中国市场来进行，这就使得当前一线城市进行国际化测试和本地化测试的需求量越来越大，要求也越来越高,因此如何适应这种来自国际市场上软件测试需求方向上的变化，还有针对这些变化进行一些国际化本地化测试的自动化，也越来越重要。

在正式进入本文所要探讨的内容之前，先对全球化,国际化测试和本地化测试相关概念做一个简单的介绍，以帮助读者能够快速了解和理解国际化和本地化测试中主要存在的 issue 和需要关注的侧重点。

主要有三个概念：全球化，国际化和本地化

全球化：简单而言，就是英语单词 Globalization 的翻译,也被称为 G11N,因为中间一共省略了 11 个单词。各个不同的大公司对于全球化的测试概念各有差异，但主要都是指软件设计开发时要注意对于多种语言和各个不同语言上的用户习惯的支持，在实际工作中，主要是包含国际化测试和本地化测试两种。

国际化：英语单词 Internationalization 的翻译，又被称为 I18N,也是因为中间省略了 18 个单词，主要是指软件设计和文档开发过程中，能够使软件主要的功能和代码，支持多种不同语言和文化；同一源代码开发出来的软件，只用再加入针对本地化的少量代码，便能够很快翻译成不同的软件版本，而不需要在代码功能设计层面做太多的改动。在做国际化软件测试工程中，经常会遇到的问题就是软件中的硬编码和一些输入缓冲区没有对本地化的支持，还有 Unicode 字符集的问题，以及字体显示在各种不同语言平台上的表现。

本地化：英语单词 Localization 的翻译，也被称为 L10N,中间省略了 10 个单词。就是将通过国家化测试后的软件，再翻译到具体的语言平台上，这一过程中，主要包括软件中的文字翻译问题，还有就是在将国际化的软件翻译后，部署到具体的本地化软件平台上，针对具体的平台和特定的语言，支持本地化的风俗，习惯等等。譬如希伯来语和阿拉伯语的输入方式就是自右

向左。

而针对软件的国际化和本地化的测试过程，便被称为软件国际化测试和本地化测试。在这两种测试过程中，通常会面临着项目测试周期短，测试人员不足而测试覆盖面大的问题。为了解决这种问题，选择一些适合进行自动化测试工具，通过自动化测试来减轻手动测试的工作量就显得顺理成章了。

而通常进行国际化本地化测试的时候，有相当大的工作量是针对用户图形界面，即 GUI 的测试，主要检查点就是一些常见的如菜单，按钮，输入文本框，复选框，JS 和 Ajax 等等，针对这一部分的自动化测试，当前市场比较流行的有几种框架，像基于 web 的 ruby+watir 框架,selenium+ruby/java/python/C 框架，还有 QTP 的一整套基于 web 的框架和对象库。

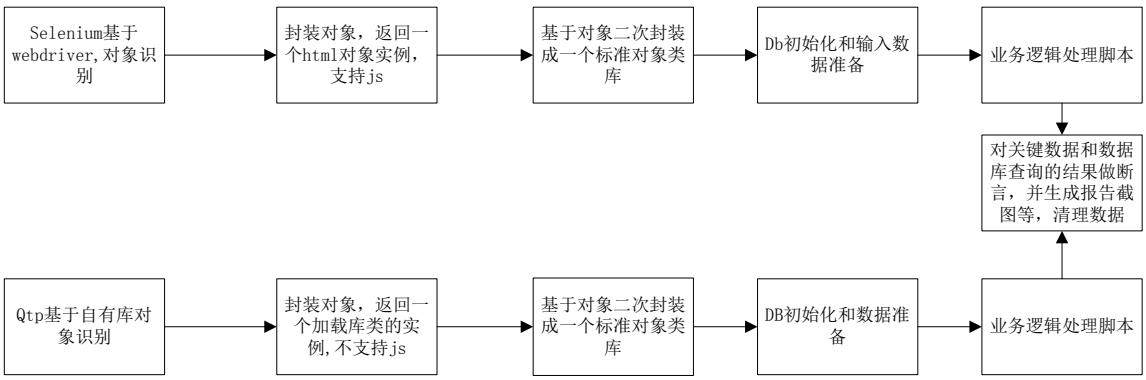
本质上，在针对 web 测试的时候，这几种框架的思想都是大同小异，都是一种基于关键字驱动的测试思想，也就是说，将 web 页面上的相关的那些 html 元素，如输入文本框，按钮，菜单等等都识别成一个个的对象，并保存到相关对象库中，然后再对这些识别出来的对象做二次操作，开发脚本。

主要的不同在于:

1.对象识别机制上，selenium2.0 后的版本集成了 webdriver,它的识别对象机制是通过解析 html 文件的 dom 树，可以通过对象的 id，classname 等 html 属性,以及浏览器提供的 xpath 机制来找到对象，而 qtp 则是采用其本身固有的库函数和类，来将页面元素映射成对应的类对象,而 ruby watir 的对象识别机制近似于 selenium.

2.对于 js 和 ajax 的一些处理，selenium 是支持对 js 的处理，而 qtp 一般地很少涉及到中间动态的过程的处理，只关心最终结果，这使得 seleniu 在处理需要关注前端 js 和 ajax 处理场景的时候比 qtp 更显得灵活

但大体原理上都是类似如下图:



通常，都是先识别对象，再对已经识别好的对象做二次封装，然后再对封装好的对象用一些简单的描叙性语言如 ruby,vb 来串通流程,再针对数据初始化写一些数据库的开启，关闭，查询等功能数据库函数做准备，准备好数据后，开始执行脚本处理业务逻辑，处理完业务对数据做校验，并清理数据库数据。

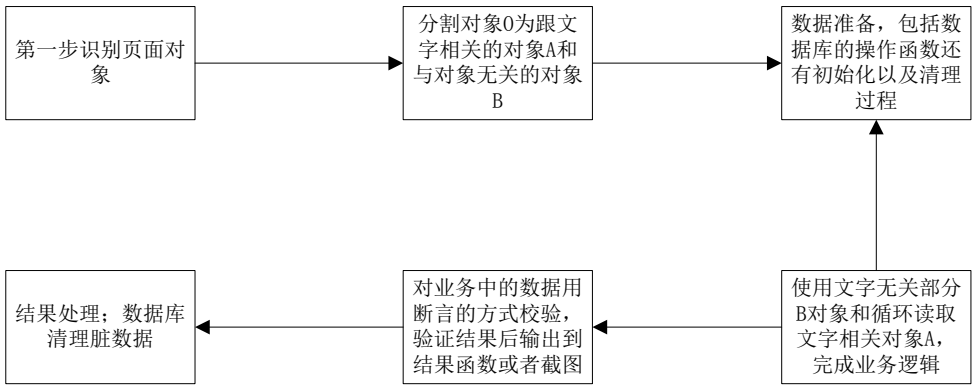
譬如典型的流程如下，用户登录，输入用户名，密码，点击 ok,提交表单，表单验证后，对后续业务逻辑进行处理，处理完业务逻辑，查询收据库数据做一个断言，然后清理输入数据。

实际测试过程中，各种测试框架各有优劣，本文也就不一一指出。本文主要关注的是国际化本地化测试过程中自动化测试框架的设计思想和具体技术的实现，主要介绍一种如何利用 qtp 的各种对象识别机制，实现国际化本地化测试的自动化测试框架方法。

如果仅仅是一个基于 web 的英文版软件，或者是特定语言的软件，那么针对他们的 GUI 测试，实际上只要识别出相关对象，修改好保存在对象库中的对象，能够保证脚本在回放过程中，脚本能够调用对象库的方法驱动实际对象就行。这个过程中最大的难点是在于对象的识别和修改，以及程序和软件升级后，如何维护这一对象库。

而在进行本地化国际化测试过程中，相比较于单一语言，更大的挑战在于同一个脚本能够支持多个不同的语言版本，也就是说，基于英文版开发出来的脚本不但支持英文版，而且还要在整个 web 页面发生变化后，只需要少量的改动，脚本还要能够同时支持日文、中文或者更多语言版本。笔者在进行这一自动化测试过程中，发现最大的麻烦还是在于对象的识别和修改上。

考虑到 qtp 强大的兼容性，还有灵活的对象识别机制，作者的主要测试框架简单如下图所示，



第一步先识别对象，这个过程中，不同语言版本的对象在用 qtp 识别的时候，往往会被映射成不同的对象库对象，所以这个时候最重要的操作就是要将对象的与语言文字相关的属性剥离出来，可以新封装一个类，然后在类里面使用这个对象的与文字无关的部分，在处理页面逻辑时，采用新类代替。

第二步在处理业务逻辑的时候，做个开关或者通过 switch 语句孔子，根据不同的语言，选择不同的文字描叙，在脚本里拼接两个对象。

第三步，是数据处理，一般地在处理业务逻辑之前，最好是预先把跟数据库打交道的 sql 查询语句还有数据库连接、关闭以及插入、删除等封装成函数，在处理业务逻辑的时候，用断言来响应数据库查询的结果和页面执行时的结果。

第四步，是对业务完成的处理，通常对各个断言语句加上输出语句，然后直接调用截图函数或者生成报告。

以上便是主要的针对 web 进行 gui 测试，特别是国际化本地化测试时，笔者所采用的自动化测试框架。当然根据项目大小不同，可以灵活采用不同的对象识别和封装机制，譬如也可以将描叙性文字剥离到一个资源文件里面，每次执行脚本的时候，通过读资源文件来处理。■

调试基于风险的测试

——对四个常见问题的解决方案

◆译者：于芳

基于风险来测试是一项技能。要了解一个产品可能出现问题的方面好，决定这些失败如果发生会产生多重要的影响，然后开发和执行测试来发现他们不管这个产品是不是在这些方面上出现了问题并不简单。当实施与典型的开发环境中，加上最后期限即将到来的压力，缺乏完整的信息和对于产品上哪些地方出错的不同观点的存在让这个过程甚至更难。但是这是一项值得学的技能。对风险关注将会把测试的关注点放在通常是测试任务的核心：快速找出重要的问题。

在我的咨询案例中，我注意到一些特定的常见的问题，这些问题是测试人员和测试经理在追求基于风险的方法时经历过的。我们来看一下这四个问题，看看怎么克服他们。

问题 1: “每个人都认为基于风险的测试只是一个管理问题。”

我经常听到有人谈到基于风险的测试时，他们看起来像是只讨论了基于风险的测试管理---根据特定产品的风险分配资源到测试活动。基于风险的测试管理的首要问题是测试活动是最有用的，和我们应当做其中的多少部分。举例来说，你可能使用风险作为决定的基础让四个测试人员来测试一个网站的性能和可扩展性，而只让一个人做功能测试。很酷。但是测试人员被指派给其各自的任务后会发生什么呢？难道基于风险的测试设计没有价值吗？

解决方案 1: 也使用风险来驱动测试设计

当我使用基于风险的测试这个词时，我的意思是包括基于风险的测试管理和基于风险的测试设计两部分。测试设计紧随测试管理其后。基于风险的测试设计是一个披露特定产品风险信息的设计测试的过程。基于风险的测试设计的首要问题是该产品是否会在我们担心他可能会出问题的样子/方向上出问题。结果是一组特定的测试来决定。

解决这种任务的一个方法是创建一个风险类（也称作缺陷分类）。在其最简单的表格上，这

是一类可能会在产品中出现的缺陷列表。我指向清楚地说“缺陷“，意指任何会威胁到产品价值的东西。它可以包括错误，失败，或威胁条件。当我写风险类时，它是一个“有东西可能出错”的表单上的声明的大纲。如果我有更多的信息，如什么导致了该问题的发生或者该问题如果发生的话会有什么影响，我也许也会把它包括进来。我已经做好了风险类别，用了一整页来描述每个缺陷。

但是一般来说，我更喜欢用一个单独的句子或句子片段来表述那些风险分类说明。

凯姆.卡訥发表了一篇相当全面的风险目录分类在其书《测试计算机软件》中作为附录。但是那个目录在 web 时代到来之前就存在了。现在凯姆的一个在佛罗里达科技的学生，吉瑞.维佳阿拉格哈文，已将所有测试电子商务购物车有用的风险目录类放到了一起。

吉瑞做的列表包括下面项:

- 购物车所在数据库的系统内存不足

硬盘失败/硬驱动崩溃，和其他可能造成数据丢失的购物车数据库的不可恢复媒介的毁坏

- 购物车数据库备份毁坏

吉瑞列表中的所有东西都于你可能要去测试购物车功能的事情有关。这个关联不是直接的。这不是一个测试清单。但是对每个他列在清单上的项，我们可以问自己一个问题“我可以执行哪类测试将会发现与其相关的问题？“这个列表写得很详细，只需要花费少量时间就能想到具体的测试用例。

我担心的是如果我以来其他人的风险目录，我可能不能充分发挥我的聪明才智。因此，如果我在做基于风险的购物车功能的 测试设计，我将不能在一开始就使用吉瑞的分类。相反地，我要让自己熟悉功能集，花上几小时，独自思考其中的风险。我要制作自己的列表。然后拿出吉瑞的列表，对照着我的列表，看是否有某些重要的东西被我遗漏掉。通过这样做，我获取到了吉瑞列表的所有价值，我可以对该风险列表负担起更多责任，我也肯那个看到吉瑞列表没有给出的一些风险。

一旦你有了一个风险目录，你就可以决定哪些项值得去测。然后对每个项，设计能够探寻到回答这些问题的测试用例—这个问题会发生在我的产品中吗？如果是的话，该问题会有多糟糕？

问题 2: “我的风险列表乱作一团”

有很多识别和分析风险的方法。我要讲一下风险分析的推理过程和他们怎样让好点子明朗起来的。但是当你与其他人这么做的时候会发生什么？然后这不不再仅仅是一个推理的过程，却也是一个社会化的过程。不同的人对该情景的表达不一样。想法常常重叠互相冲突。

我的同事布来特.皮提考德在最近的一个项目中经历过这样的事情。“我们与测试团队在会议上开始讨论风险列表”，他回想到，“我们有 6 个人，很快地产生了一个长长的风险列表。难的是找出怎样管理这个列表。我们要担心这个长长的列表上的东西，但是对很多项来说，我们并不十分确定要做什么。”

就像他所说的，“我们的很多项都是模糊的。一个项是“临近干扰”，意思是添加到该列表上的某些东西，但是当我们晚点再看它的时候，我们记不得了。这是什么意思？我们也有很多不同类型的事情在任务列表上出现。我们列表上的风险包括风险，也包括测试类型（如负载测试），产品性能和质量标准（如可扩展性或安全性）。我们需要一个对一个风险是什么样以及其与将要出现的其他一些问题是怎样关联的有个更精确的理解。”布莱特在这里是广义地使用“风险”这个词。

既然测试是有关发现产品中的问题，基于风险的测试是有关产品的风险。布莱特的团队列出了很多除了产品风险外的东西。现在，这是一个开头吧。头脑风暴中头脑中观点乱七八糟混作一团很正常。实际上这是一件好事。头脑风暴的核心目的在于最小化重要想法没有记录下来的几率，即使有一百个奇怪的点子也同时出现。问题是，你不能拿头脑风波的原始输出期待将他正好放入到测试计划中。“我们习惯于使用能给你一个要做事情列表的测试或缺陷列表”，布莱特说，“这样当你完成任务时，所有东西都将被一一核清。我们现难以排出怎样核清我们清单上的东西。”他的挑战是将该列表转换成某些可行的东西。

解决方案 2a: 以类型归类

我发现有一件事很有帮助，它是为在风险头脑风暴中想到的不同事情做不同的列表。基本上来说，我想让在每个单独列表上的所有东西与一个单一种活动相关联。那样的话，我可以捡起一个列表来工作，而不用变得分散和迷惑。

我使用下面这样的列表:

产品风险列表:

可能在产品中发生的问题（如“web 服务器在正常负载下可能有点过慢”）。一个产品风险能够激发出一个测试或测试技能。

任何具体的产品需求都可以在风险分析中重新变成一个风险。对任何需求来说，有一个与不能达到需求要求相关联的风险。如果一个特定的需求只有很少的相关联的风险，那么要么是不把他列为风险，要么把他与其他类似值得列出的风险归到一起。

对此你要做什么：

对列表上的每个项，你做以下任务中的一个：

- 创建测试用例评估风险
- 深入调查风险（不必要去测试它）
- 接受该风险（完全不测试它）
- 将该风险转移给其他人（或许是让开发人员通过重新设计该产品来减少风险）

随着项目的进行，你使用该产品风险列表作为汇报问题的基本来源。在一个运行良好的项目末期，你的测试将披露足够的信息给予管理层信心每个风险的状态都是已知的。

风险因素列表：

倾向于创建增加的产品风险的条件（如开发以前没有跟这个 web 服务器工作过）。你不是测试风险因素；你测试的是风险因素的存在。举例来说，“功能 X 十分复杂”并不表示有一个特定的问题，你要尝试在产品中去发现，它仅仅表示任何种类的问题都更可能在复杂的怠慢中出现。另外一个风险因素种类是一个“威胁”---一个条件或输入以某种方式给产品施压，并且潜在地开发了一个脆弱性。要决定一个风险想法是一个风险因素还是一个产品风险，问自己如果你把它报为一个缺陷时会发生什么。如果回答显然是“不是一个缺陷”那么它就可能是一个风险因素，但可能不是一个产品风险。

你要做什么：

既然风险因素是造成产品风险的东西，你可以使用本列表上的项来帮你找到产品风险可能是什么及其潜在的严重性。他们也可能会帮你想到风险降低策略（尽管那可能超过测试的范畴），可测性要求你会想要与编程人员来往或者执行具体的测试。举个例子来说，“破坏的客户平台”可能是在一次风险头脑风暴里想到的风险因子。我会将这个风险因子放在风险因素列表上，然后问自己这样的问题—什么样的产品失败会更可能发生在破坏或者配置不当的客户平台上？我们将怎么检测到这样一种情形？我们怎样创造那样的情形？

风险区域列表：

类似产品风险归组（如性能）。一个风险区域意味着至少有一个产品风险，而且很可能更多属于产品风险。在风险分析过程中，一个基本的质量标准目录，如兼容性或性能，可以认为是一个风险区域。换句话说，没有达到兼容性标准的风险可能指引我们去包括“兼容性”（或“非兼容性”，如果你更喜欢这种说法的话）

作为一个风险区域。

你要做什么：

这个列表上的项可能用来作为基于风险的测试计划的头条内容，下面归类着具体的风险。

风险区域帮你总结归纳风险及其相关联的测试策略。对任何风险区域，你都应该能够去想象各种产品可能会崩溃的具体的方式。

如果你不能，那么或者这实际上并不是一个风险区域，或者你需要收集更多的信息。

问题和问题清单：

一个大包揽，包含了所有需要调查或者回答的东西，让项目得以继续进行（例如，我们购买负载测试工具的预算够吗？）

你要怎么做：

解决该问题，升级上报或者忽视掉。这是标准的项目管理的东西。

产品组件或功能列表：

这些是被认为与一些升级的风险（如 web 服务器）相关联的产品部分。一个特定的组件也许有一整套与之相关的复杂风险组合。试着把这些风险变成公开的东西。

你要做什么：

你可以使用这个列表在基于风险的测试管理中来分配对产品的注意力。我喜欢将产品组进一个用一页或两页纸正好描述完的组件列表，然后把每块区域标成需要“较高”，“普通”或“较低”级别的资源。当一个产品组件出现在风险头脑风暴里，你可以讨论是否它仅仅是一个“普通”的风险（通常，我会假设所有的东西都需要测试），或者一个相较于其他组件是较高级别的风险。

测试思想列表：

测试建议（如负载测试）。测试思想值得捕捉，因为晚点当你在弄清楚怎样以能够描述这些

风险的方式去测。当测试的思想在风险分析中出现的时候，又他们看起来很重要的话，问自己什么具体 的风险可以把他们激发出来。

你要做什么：

本列表上的项最终会形成测试策略或特定的测试用例。

项目风险列表：

影响项目成功但不一定会影响产品操作的风险（如需求文档没发送）。

你要做什么：

这很可能不是你的工作，作为一个测试机构来运行项目或者做质量警察。

但是，当头脑风暴里出现有关项目风险的项目风险时，要把他们记录下来。然后，再来看那个列表上的项，决定哪些风险威胁项目的测试部分（哪些对测试经理来讲成为一个问题列表的风险），哪些风险威胁到项目的其他部分（哪些你你向项目经理或其他人员吐露的风险）。

风险降低措施：

这是你想做以降低风险的事情（如限制对生产服务器的访问）。他们也许超出了测试项目的范围。

你要做什么：

直到你对该项目有掌控权，你才可能将这些想法传达给项目经理。这不是一个目录消耗列表；我不确定有这样一个目录。如果你发现其他一些信息类在你的风险头脑风暴里出现，而你又不能将他们简单地塞进这些类别中，那就做一个新列表。对任何你创建的新表，回答这个问题“我要拿这个列表做什么?”否则，就不要做这个表。

你可能在头脑风暴里做了这些表，并且其后将这些项归类，或者只对一个种类的项做头脑风暴。当然，你也可以把所有的东西都乱糟糟地放在一个大的 ol 表上，只要你知道要怎么对每一种想法。

解决方案 2b：认识你的参考框架

记住什么是风险。我喜欢这样的定义：有坏事要发生的危险。对任何产品风险，我会立即想要知道这些东西：“那些坏事是哪些事情会发生？发生了结果会有多糟？这事情发生的可能性有多大？“这些问题需要我们来决定因果。风险表变得混乱的一个原因是人们在原因和效果的

迷宫中迷失。

拿“不正确的输入”做个例子。它是一个原因还是一个结果?不正确的输入是可以引起产品出现失败的原因还是它是设计糟糕的用户界面的结果？或者不正确的输入是由于另一个已经失败的子系统的 输出导致的？看你的参考框架，这可能也可能不是一个产品风险。这里没有一个正确的答案。用户体验作为一个产品是一个长期和复杂的因果关系链。我们都跟着那个链测试。

一个保存你的支撑物的方式是想到这个基本连锁:

受害者<-问题<-脆弱性<-威胁

受害者：经历一个问题的影响的人。最终没有缺陷会是很重要的指导它危害到一个人。

问题：产品做的我们希望它不要发生的一些事。(你也可以称这个为“失败”，但是我能想象到问题不是失败，严格意义上讲。)

脆弱性：有关该产品的引起或者允许其在一定的条件下显示出问题的东西（也称作故障）。

威胁:

产品外部的一些条件或输入，如果发生的话，将会造成脆弱产品上的问题。

对于这个链，我们可以说因为在操作产品时，由于产品中的某些脆弱的点被一些威胁利用可能会有事，有人可能会因此受伤或受到干扰。这基本上是一个有关风险的迷你故事。不管你想到什么样的风险思想，在故事中找到它的位置，然后试着去将故事的其他部分变得有血有肉起来。因此，你需要设置你的参考框架：首先决定你在谈论的什么产品或子系统—是什么弱点，直面威胁，显露问题并且影响到受害者的?

问题 3:“我项目上没有人想谈论风险。”

风险是很可怕的。它是情绪化的。如果你是一个刚刚成功地为使用一种新的、未曾尝试过的方法来设计的设计师，就要十分警惕风险。

例子

我们假设你的团队主持了一次风险的头脑风暴，结果是有一个长长的有很多东西的表，如下:

1. 性能和可用性

2. 十分大的业务量

3. 我们对 web 服务了解不足

4. 自动化回归测试在双曲线树组件上不可能。还有我们对源代码没有控制权。

5. Web 服务器崩溃

花点时间试着将这些东西做归类。

这里是我的想法:

性能和可用性风险区域。下一步是获取每个风险区域的更详尽的东西。

十分大量的业务量可以是风险驱动器，测试思想，产品风险或者产品的功能。谁知道？这个需要更多的上下文背景。

我们对 web 服务器了解不足!! 听起来像是一个某些人认为是相当重要的一个风险因素。其实它也可能是一个问题，一个项目上的风险，和一个产品的功能。请多一些信息。

自动化回归测试在夸张的树型组件上不可能实施。我们对其源代码也没有控制权。两个单独的事件卡在一起了：第一个看起来像是一个问题或者一个项目风险，第二个看起来是一个项目风险而且可能是一个产品风险要素。放在一起，这两个事件也意味着需要对他们给予特别的关注以恰当地测试双曲线树。(如果我们决定测试所有的组件很重要，那它就是重要的。)

Web 服务器崩溃可能意味着任何事情。在根据项目前后背景能得出显然的 web 服务器崩溃意味着什么，这么说都只是很模糊的说法。

我相信这些东西的大部分都会从一些扩展或背景设置中获益。有些情况下，在项目团队里讨论他们可能造成我们 的表上又增加了很多具体的新的项，和测试什么的更好的想法。

由于项目中的技术，那么你可能不想公开承认你的想法是有风险的。如果你是一个顾客，可能你不会喜欢听到说那个你即将要使用的产品到处是风险的。如果你是一个公司律师，可能你会对周围的风险表的责任感到紧张，这些风险应当去传召法庭传票。我看到过高级经理们期望并且认为所有的风险都是“被管理的”，意即真的被消除掉。

让人们坦率地谈论风险的确是件难事。或者如果他们坦率地谈论了，那让他们去用具体和理性的术语来谈还是困难的。你可能因为“消极”被谴责过，意味着想寻找问题本身这个想法给项目带来了问题。

解决方法 3：将注意力从风险转移到选择上

你不可能所有的东西都测试得一样程度。而且即使你可以，那代价也太高了。这是基于风险测试的测试员的集体呼声。

让我们做些明智的选择。

如果项目上的人不想让你谈论风险，那就不谈。开始谈选择吧。没有人能逃脱不选择。我们应当测什么？强度多大？为什么这里要比那里测得多？不需要说出风险这个词，你就能让人们谈论它。

另外一个策略是私下做你自己的风险分析，庵后宣布你对什么需要测试的选项。仔细聆听管理层和开发者的反应。

这有点像是一个“石头汤”的办法---通过提出要给他们上一个石头，也许你会让其他人贡献一些汤。（如果你的参考书中没有这个故事的话，你可以在线读一下寓言故事。到 Sticky Notes 上找一个链接。）提出我自己的风险表很少让人们不跟我说话，如果有只是反对我的选择。当他们参与谈论，我得到了更多有关风险的信息。

问题 4: ”我不知道那不是一个风险。那样做让它变成一个风险了吗？”

每个你找出的风险都是从一个小不点除了流言开始的。在测试之前，你不知道产品中有什么样的缺陷在里面。你甚至不知道任一个特定的缺陷出现的可能性有多大。也许你知道一定的问题是可能有的。道理就在这里。当我们做风险分析的时候这让生活对我们来说变得艰难，因为不确定性可以让我们感觉是产品风险。

但是在你知道某事是一个风险和某事你恐怕可能是一个风险之间有着很大的区别。这种差别是，因为你看见一只大黄蜂在你的卧室里打转而知道自己处于被大黄蜂刺蛰的危险中和因为你还没有搜遍每个大黄蜂可能藏着的地方而不知道自己没有处在被刺蛰的危险中的差别。

为什么这个很重要？因为他对帮助我们尝试理解哪些缺陷是严重的哪些缺陷是仅仅温和的。当你尝试将你知之甚少的风险的重要性和你知之较多的风险的重要性做对比，有一个倾向将那些你知之甚少的风险的重要性夸大的可能。那样就会降低整个风险分析的可信度了。

解决方法 4：区别已通报的风险和未通报的风险

当我对产品风险归类的时候，我将他们归类成两大主要维度：重要性和信息层级。我最常用来衡量信息的尺度是很多，一些，很少和无。（我喜欢低调做事。）

通过用信息层级来区别风险，我们提供了一个重要的角度供人们来评审这个表。我们对自己对已通告的风险然后是未通告的风险的评估也更加自信。如果我们说一个风险是严重的，但是我们对之知之很少，我们基本上会说有一些需要考虑的原因。通过调查那个风险，包括对其进行测试，我们可以得到信息。我们作为测试者的目标是将所有重要的风险转换成完好描述确认的风险。这样做可能会让我们去发现一个给定的风险并不是如那么描述重大的风险。或者业务我们确认了一个风险是严重的，这激励管理层和开发者采取行动来提高产品质量降低那个风险。

已揭露！基于风险的测试的伟大秘密对具体的问题如此重要，现在更是对一个有用的可以斩断基于风险的测试的整个频谱的观点非常重要:

差劲的缺陷分析是一个自我纠正的问题。这是真的因为如果你的 测试不是足够的基于风险的，那么你 可能会与重要的问题一起沉没。 然后你就会说“呼啊，下次让我们多关注些那样的问题。”那一轮学习是从桥梁（见做工程要讲人性，亨利.派特若思基著）到飞机（读任何 NTSB 飞机事故报道）做工程的一个基本部分。那是为什么你不应当太苛求自己当你在一个新的、你并没有工作经验的产品线上使用基于风险的测试方法时的原因。有些风险你能够预测到。其他的你不能。没有方法可以提前确定这些风险都是什么。任何人从你这里期待得到的所有东西是深思熟虑并且富有资源，让人们对你的分析的限制有所了解，并且从经验中学习。

那些避开我们的测试网的故障将界定我们将来应当怎么测试---如果我们在注意这个问题的话。只要记住马克.吐温对此说的话:“我们当小心走出一种只有智慧在其中的经验本身—并且停在那里;至少我们会像那个坐在热火炉盖子上的那只猫一样。它再也不会坐在热火炉盖子上---那样很好;而且它也再不会坐在一个冷炉子盖上。” ■

网站导航检查的自动化实现

作者：郝强

本文讲述了如何实现网站导航检查几种思路其中还包括对于需要登录网站进行抓取也提供了一些指导。

1. 网站爬虫与导航检查?

对于互联网企业以及有互联网产品的企业，总会有类似这样的需求，你需要对你的全部站点的链接进行检测，以确保没有死链，这是基本需求。此外，有些互联网企业要求自己的网站在每次发布之后，能够把所有链接被爬虫爬一次（访问一次），以保证那些需要动态生成的页面能够生成静态页面或缓存。更有甚者，有些公司希望能够对每个页面的一些元素进行检查，以保证页面没有错误。

目前，大多数企业多已经有了自己的自动化测试，类似使用 qtp, selenium 以及 watir 等，所以大家可能会期望使用这些工具在实现上述需求。但是，上述需求的验证工作大多数都有时间要求，通常希望能够在最短的时间内完成检查。通常情况下是最长时间不能够超过一小时。所以在这里我将和大家一起讨论用一些其它的技术来实现上述需求。

2. 如何实现它

首先我们考虑用 ruby 来实现，为什么用 ruby 呢，主要原因是我还比较熟悉。呵呵，开个玩笑，我推荐大家使用 ruby,相信你会着迷的。而这里我们主要基于用 Nokogiri 库来实现。 Nokogiri 是一款用 ruby 实现的 HTML,XML 以及 SAX 的解析库。它能够通过 xpath 或 css3 的 selector 在文档中进行搜索。

为了实现之前我们提到的需求，我们首先考虑要给程序提供一个入口，即爬虫从哪里开始爬，第二点我们的程序应该有能力去掉一些不想爬的页面，最后呢因为有时间要求，它应该是个多线程的程序。

关于入口我们将其存储在一个配置文件中，在 ruby 中我们选用 yaml 格式，配置文件的读取基本实现类似下列代码:

```
require "yaml"
require "net/http"
require "uri"
require "net/smtp"
#require "mailfactory"

module Configuration

  HOSTS = File.dirname(__FILE__) + "/hosts.yaml"
  MAILERS = File.dirname(__FILE__) + "/toList.yaml"
  THREADS = 10
 # RESUTLS_PATH = File.dirname(__FILE__) + "/../Results/"
  CONCURRENCY = 3
end
```

上这代码主要定义一个入口配置文件 hosts.yaml 以及一个邮件通过列表 toList.yaml，以实现在运行结束后能够发送邮件报告通知给指定的人群。

当我们的程序开始运行时，它首先从配置文件中将入口地址读入，然后解析它并对应到相应的二级域名上。类似实现代码如下：

```
#parse the link and generate all links which are domained
def parse_link(url)
  begin
  doc = Nokogiri::HTML(open(url).read.strip)
  # get all links on the page
  doc.css('a').each do |link|
    @all_links << "#{link['href']}" if link['href'].to_s.include?('http://www.[redacted].cn')
    @all_links << "#{link['href']}" if link['href'].to_s.include?('http://[redacted].com.cn')
    @all_links << "#{link['href']}" if link['href'].to_s.include?('http://www.[redacted].com')
  end
  rescue
    puts url + " maybe down!"
  end
end
```

有些链接是我们不需要检查的，所以爬虫不应该去爬它，我们提供一个方法能够去掉它，类似实现如下所示：

```
# delete all links which are not need to accessed.
def remove_notused
  tmpdata =[]
  @all_links.each do |link|
    tmpdata <<link if link.include?('myOrders.jsp')
    tmpdata <<link if link.include?('profileHome.jsp')
    tmpdata <<link if link.include?('[redacted]yCoupon.jsp')
    tmpdata <<link if link.include?('[redacted]extHelpcenter.jsp')
    tmpdata <<link if link.include?('[redacted]pClassification.jsp')
    tmpdata <<link if link.include?('myFavorites.jsp')
    tmpdata <<link if link.include?('customer.jsp')
    tmpdata <<link if link.include?('groupCouponlist.jsp')
    tmpdata <<link if link.include?('myComplaint.jsp')
    tmpdata <<link if link.include?('gomeNewsAllShow.jsp')
    tmpdata <<link if link.include?('link.jsp')
    tmpdata <<link if link.include?('accountSecurity.jsp')
    tmpdata <<link if link.include?('surveyPage.jsp')
    tmpdata <<link if link.include?('cart.jsp')
  end

  tmpdata.flatten!
  tmpdata.uniq!
  tmpdata.each do |data|
  @all_links.delete(data)
  end
 # puts "Removing below links which are not need to be tested!"
 # puts tmpdata
end
```

其它我们需要将已经生成的域列表对应到内部各个集群的具体实例上，也就是内部的 ip 地址上。最后需要所有列表组合在一起生成一个大表。

```
# get all instance data
def get_cluster_data
@hosts.each do |key,value|
  @glist << value if key.to_s.include?('cluster1')
  @glist << value if key.to_s.include?('cluster2')
  @[illegible]list << value if key.to_s.include?('cluster4')
  @[illegible]list << value if key.to_s.include?('cluster5')
  @[illegible]list << value if key.to_s.include?('cluster3')
 end
  @glist.flatten!
  @g[illegible]list.flatten!
  @cc[illegible]list.flatten!
end

def get_instanace_count
  @glist.size + @[illegible]list.size + @cc[illegible]list.size
end
```

生成列表后，如果你的 web 服务器有开启 rewrite 而生成伪静态，那么你还需要对你的列表进行处理。这里不打算详细讲述这一部分，主要是将 url 用正规表达式转换到对应的伪静态地址上。

为了能够实现多线程，我们在实现此程序时使用的时 ruby 1.9.3，我们选择将网站爬虫要爬的 URL 列表存储在 queue 中，因为在 ruby 1.9 中,queue 是线程安全的。然后我们得有能力让线程发送请求到指定 URL 上。注：本程序并没有实现解析部分。

```
def arrange_url
  #puts @runList
  @runList.each do |url|
    @queue << url
  end
  puts @queue
end

# send request to instance wihich will be called in thread
def send_request(link)
    begin
    puts link + " " + Net::HTTP.get_response(URI(link)).code
    puts link
    rescue
     puts link + " maybe down."
    end
end
```

可以看的到，我们会取到 web 服务器返回的 http 状态码。

那线程到底是怎么实现的呢，我们参考如下示例代码：

```
Configuration::THREADS.times do
@threadgroup << Thread.new {
  i.times do
    if !@queue.empty?
      t = @queue.pop
      send_request(t)
    else
      puts Thread.current.to_s + " is running stop..."
      Thread.exit
    end
  end
  puts Thread.current.to_s + " is running stop..."
 }
end

@threadgroup.each do |tg|
tg.join
end

puts "Total Run: " + (Time.now - start).to_s + " seconds!"
```

可以看得到，我们从配置文件中取得线程数并生成线程，每个线程的工作就是去 queue 上拿一个 url 下来，然后发送请求到服务器，全部结束后返回。所以大家可以看得到，整体思路就是我们先得到要抓取网站的入口地址，然后解析出对应的内网 IP 地址并生成 URL 列表，然后通过多线程去访问 URL。当然最后我们是有生成一个报告并可以邮件通知的。

3. 其它的实现方法

除了上述的自己编程及定制的实现方法外，我们也可以参考一下现成的 gem 包。我们首先发现的是 Anemone。Anemone 是一个非常容易使用的基于 ruby 的 web 爬虫框架。它主要是提供一组简单的 DSL（领域特定语言 Domain-Specific Languages）去处理网站每个页面的 Action。

那么 Anemone 提供了哪些功能呢？

1. 高性能的多线程设计
2. 跟踪 301 HTTP 转向
3. 通过内建的 BFS 算法决定页面深度
4. 允许通过正则表达式去排除基于 URL 的页面
5. 支持 https
6. 可以记录每个页面的响应时间
7. 可以记录页面深度等。

下面有个例子展示了如何使用 Anemone.

```ruby
require 'anemone'

begin
  # make sure that the first option is a URL we can crawl
  url = URI(ARGV[0])
rescue
  puts <<-INFO
Usage:
  anemone count <url>

Synopsis:
  Crawls a site starting at the given URL and outputs the total number
  of unique pages on the site.
INFO
  exit(0)
end

Anemone.crawl(url) do |anemone|
  anemone.after_crawl do |pages|
    puts pages.uniq!.size
  end
end
```

这个例子主要是展示了如何去得一个网站上页面总数。

```ruby
require 'anemone'

Anemone.crawl("http://www.example.com/") do |anemone|
  anemone.on_every_page do |page|
      puts page.url
  end
end
```

上述这段代码展示了如何打印出整个网站上的 URL 地址。这里我们不做过多讲解，大家可以去 Anomone 的官方站点多做研究，就可以基于此框架实现各种功能的爬虫。

4. 爬取需要登录的网站

看了上边的例子，大家一定都在想，如果是去爬或抓取非登录的网站，上述所讲的那些内容应该是差不多足够了，如果我们想去抓取需要登录的网站，怎么办呢？其实我们在工作中，许多企业级的 BS 应用都需要认证登录的，然后才可以操作其功能，对于这样的网站如何抓取的问题，我做了一些研究，发现也是可以的。下边我就来介绍一下。

这里主要是要用到 ruby 的 Mechanize 库。Mechanize 库是用于与 web 网站进行自动化交互。Mechanize 自动地的存储和发送 cookies,转向及提交表单。它也能跟踪你访问的站点的历史。

使用 Mechanize 抓取网页非常简单，请看示例:

```ruby
require 'rubygems'
require 'mechanize'
agent = Mechanize.new
page = agent.get('http://google.com/')
```

模拟点击事件

```
page = agent.page.link_with(:text => 'News').click
```

模拟表单提交

```
google_form = page.form('f')
google_form["q"] = 'ruby mechanize'
page = agent.submit(google_form, google_form.buttons.first)
pp page
```

大家可以从 github 上下载 Mechanize，然后参考其示例，便实现爬取需要登录网站内容。

当然，通用的思路为，我们先拿到要登录页面元素的信息，然后参加上面例子实现登录，之后再抓到所有链接进行网站的爬取工作。

通常实现登录的示例代码大致如下:

```
def login(params = {})
    @page = @agent.get(@url)
    my_form = @page.form(params[:form])
    my_form[params[:username]] = @username
    my_form[params[:password]] = @password
    @page = agent.submit(my_form)
end
```

如何调用它呢，请参考下面示例:

```
logins = {
  url: 'http://WhatYouWantToAccess.com',
  username:'yonghuoming',
  password:'woshimima',
}

forms = {
  form: 'Login',
  username: 'username',
  password:  'password',
}

test = Navigater::new(logins)
test.login(forms)
```

可以看得到，我们先定义了两个 JSON 串，一个用于存储登录信息，分别是 URL，用户名和密码。另一个用于存储登录的表单信息。然后我们实例化 Navigater 类(这里没有给出具体实现),然后调用其 login 方法实现登录。之后大家就可以在登录后利用爬虫代码实现对全站内容的验证及抓取工作了。■

简易云终端自动化测试系统设计与实现

◆作者：许松

云计算这几年已经席卷了各个角落，在 IT 行业内兴起了一股新的潮流，它必将在这几年内统领 IT 行业的发展方向。对于广大从事与 IT 行业相关的工作的人士，了解云计算的历史沿革和发展现状是十分必要的，本文首先介绍云计算的历史，云计算建立基础，云计算定义，由 NIST 规定的关于云计算的五个基本特征、三个服务模型和四个发布模型；然后分析了云计算的优缺点；接下来我们介绍了云计算的两个关键要素：云服务器和云存储，并且介绍了当今比较流行的几个云平台。最后简要介绍了一下云计算与大数据之间的关系。

一、概述

1. 背景

市面上的自动化测试基本都是基于软件的自动化测试，包括 web 软件和 GUI 软件，对于系统的自动化测试少之又少，随之云概念的普及，越来越多的云终端产品涌入市场，这些云终端的质量凸显尤其重要，基于对云终端质量的保障和减少人工失误导致误差，引入对云终端的可复用功能测试以及性能测试的自动化实现。

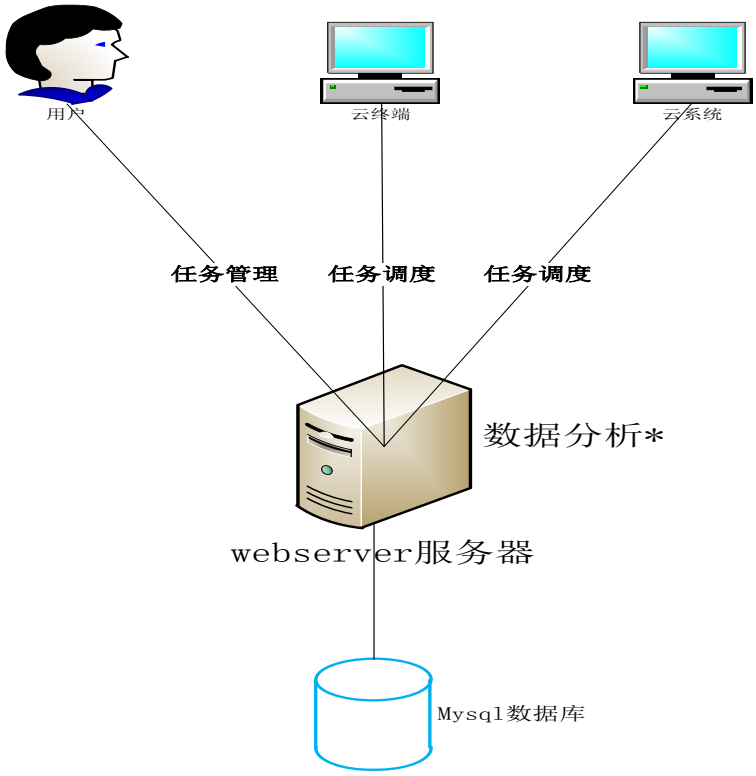
2. 术语

术语	解释
系统	本文特指支持 python 的 window 系统和 linux 系统以及基于以上两类定制的系统版本。
云终端	本文特指支持云系统，内嵌 windows 或 linux 内核系统以及以上定制版本系统的终端设备
云系统	本文特指在局域网架构下通过当前流行的远程连接协议 RDP、ICA、PCoIP 进行远程办公的操作系统。
脚本	本文特指将测试用例转换为可执行的 python 脚本
任务	每个脚本分发给终端的执行记录称为任务
前置任务	任务的执行开始必须在前置任务完成之后

二、总体设计

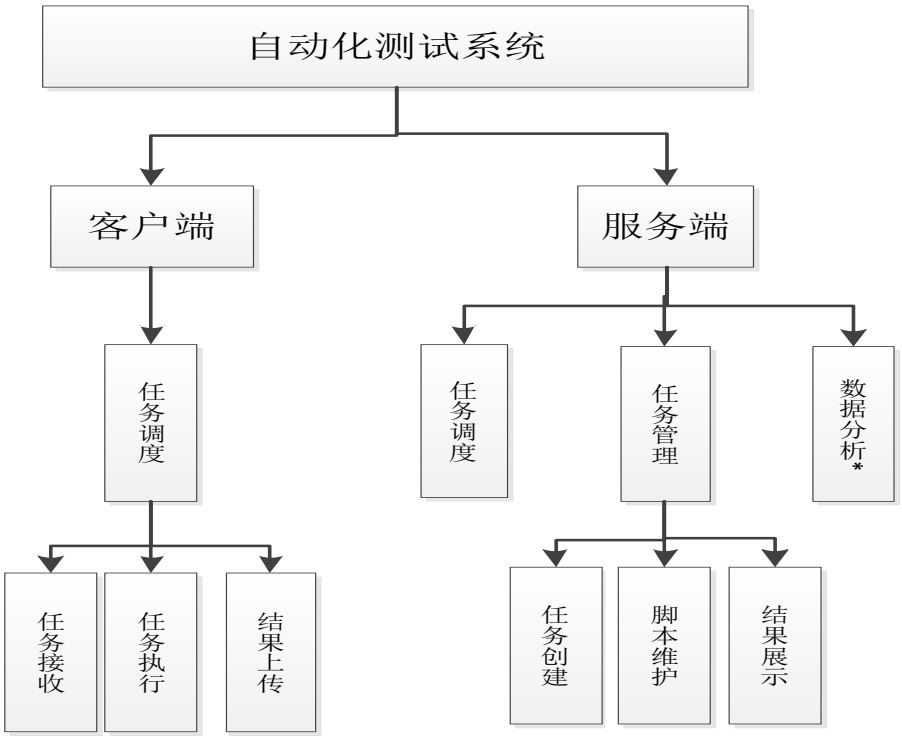
1. 角色实体行为分析

参与角色主要是：用户、云终端、云系统，参与实体主要是服务器和数据库，各个角色实体之间的行为关联如下：



2. 模块功能结构

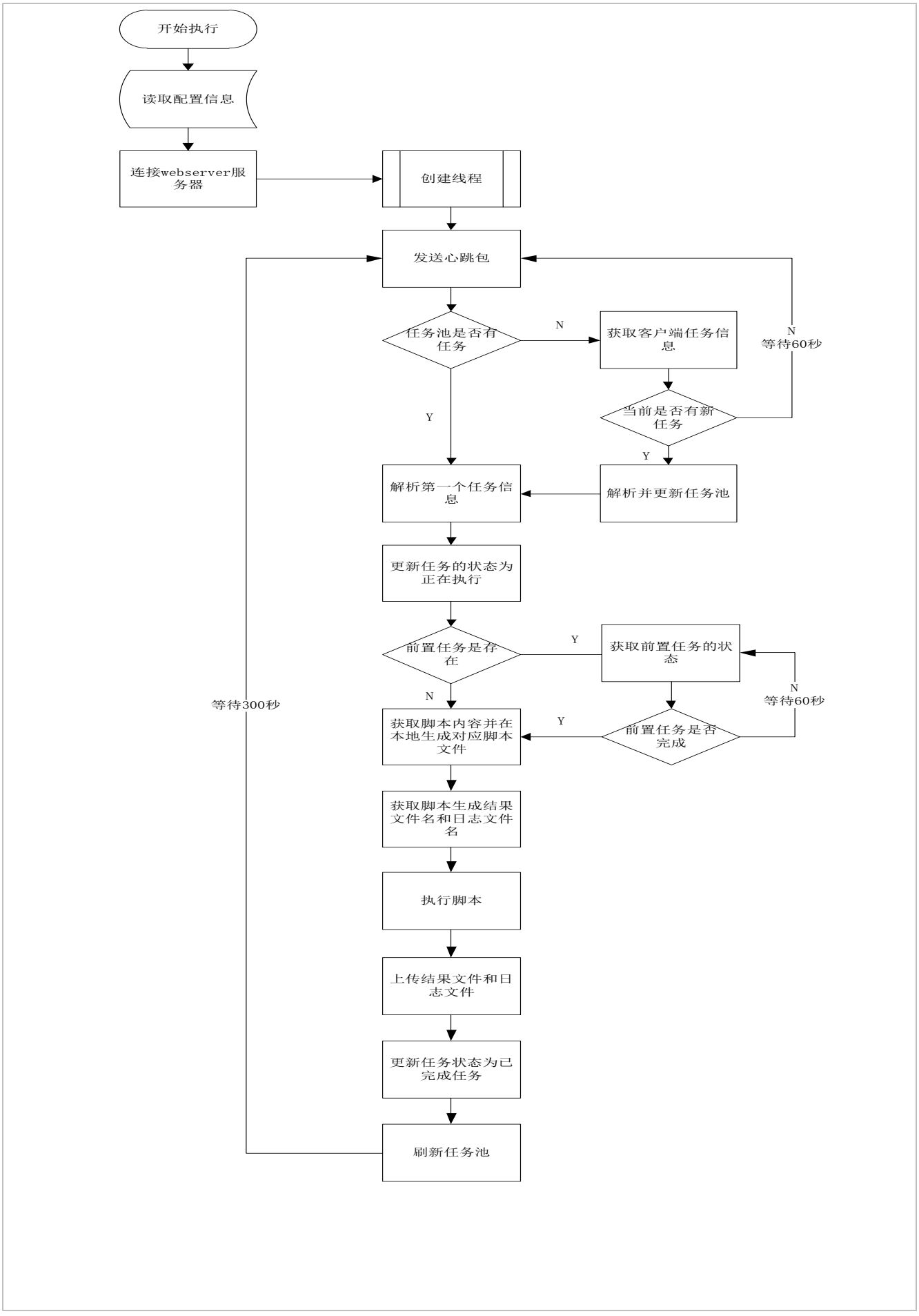
该系统设计研发主要是为云终端提供一个以任务方式下发执行的自动化测试系统，主要由任务调度和任务管理以及可自定义扩展的数据分析构成，系统结构如下：



其中任务管理面向用户，提供创建任务、脚本维护、结果查看的功能；而任务调度面向云终端和云系统，提供任务接收、任务执行、上传任务结果的功能；最后的数据分析提供可自定义扩展的测试结果数据汇总分析功能。

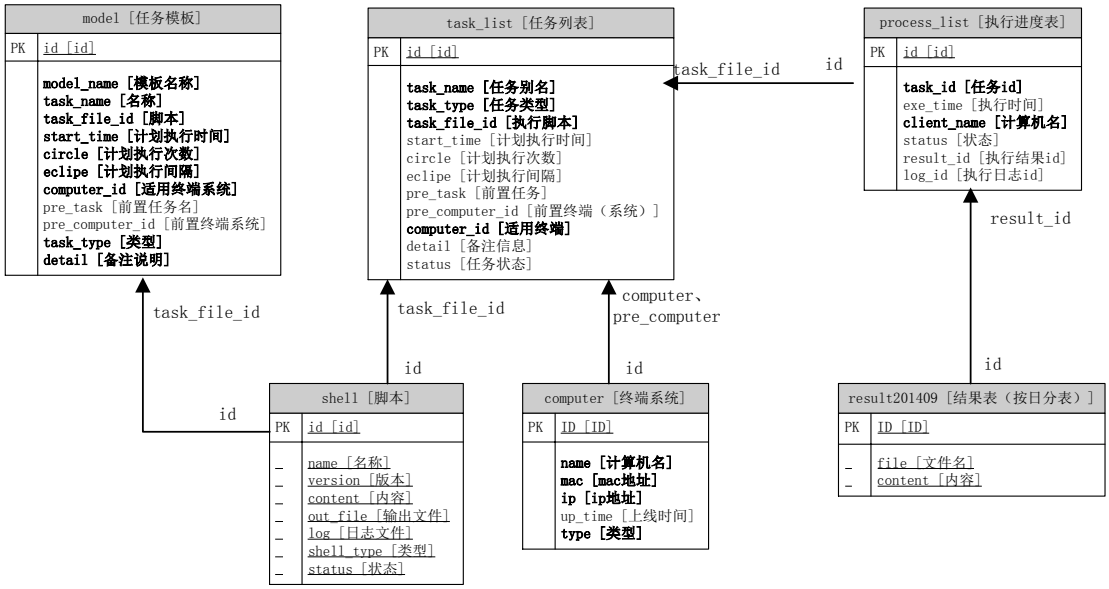
3. 核心模块流程

该框架的核心部分是任务调度，任务调度由服务端和客户端组成，客户端启动后，将所在系统信息更新到服务端，接着接收分发给自身的任务并纳入任务池，然后开始判断任务前置关系，最后执行任务，更新任务池。任务调度流程如下所示：


开始执行
读取配置信息
连接webserver服务器
创建线程
发送心跳包
任务池是否有任务
N
获取客户端任务信息
N
等待60秒
当前是否有新任务
Y
Y
解析第一个任务信息
解析并更新任务池
更新任务的状态为正在执行
前置任务是存在
Y
获取前置任务的状态
N
等待60秒
等待300秒
N
获取脚本内容并在本地生成对应脚本文件
Y
前置任务是否完成
获取脚本生成结果文件名和日志文件名
执行脚本
上传结果文件和日志文件
更新任务状态为已完成任务
刷新任务池

4. 数据库和数据结构设计

为了保证数据的可迁移以及搭建成本的降低，该该系统采用免费关系型数据库管理系统 mysql 存储各种信息。数据库设计的目的是围绕任务为核心的各种信息。



三、实现

从用户视角出发，按照任务管理->任务调度->数据分析*进行介绍实现过程。以上所有的操作都是基于 web 界面，在此笔者采用与 python 同源的 web2py 进行实现该功能模块，关于搭建 web2py 服务器具体过程，请参考 web2py 官网。

1. 任务管理

任务管理模块主要分为脚本维护、任务创建，结果展示三大功能

A. 脚本维护

脚本维护功能主要实现脚本的上传和脚本展示查看，其中脚本上传直接使用 web2py 上自带的’file’类型输入框即可实现上传功能，相关实现如下:

```
def upload():
    form = FORM(TABLE(
        TR(TD('选择上传路径:', INPUT(_type='file',
                                    _name='myfile',
                                    id='myfile',
                                    requires=IS_NOT_EMPTY()))),
        TR(TD('输出文件:',INPUT(_type='text',_id='out_file',_name='out_file'))),
```

```
                TR(TD('日志文件:',INPUT(_type='text',_id='log_file',_name='log_file'))),
                TR(TD('脚本类型:',SELECT('Custom','CCBench',_id='shell_type',_name='shell_type'))),
                TR(TD(INPUT(_type='hidden',_id='out',_name="out"))),
                TR(TD("脚本文件每次更新都将自动在版次上增加一位,默认版本为 V1.0,第二个更新文件即为 V1.1,自动递增到 V1.9 后切换为 V2.0")),
                TR(TD(INPUT(_type='submit',_value='上传'), A(T("返回"),_href="javascript:history.go(-1)",_class='btn')))
            ))
        if form.accepts(request.vars):
            ...将脚本内容插入数据库,过程省略
        return dict(form=form)
```

脚本查看直接通过 web2py 提供的 SQLFORM.grid 接口进行展示，相关实现如下:

```
def show_shell():
    query = (db.shell.status==1)
    fields = (db.shell.name,db.shell.version,db.shell.out_file,db.shell.log,db.shell.shell_type)
    headers = {'shell.name':'脚本名称','shell.version':'最新版次','shell.out_file':'输出文件','shell.log':'日志文件','shell_type':'脚本类型'}
    form = SQLFORM.grid(query=query,field_id=db.shell.id ,fields=fields,headers=headers,searchable=False,details=False,editable=True,links = [lambda row: A('详情',_href=URL("default","view",args=[row.id])),lambda row: A('删除',_href=URL("default","delete_shell",args=[row.id]))],maxtextlength=64, paginate = 20)
    return dict(form=form
```

B. 任务创建

任务创建功能主要实现任务的创建、任务展示。

在实际测试过程中，大部分测试任务可以按模块进行划分，进行大量的复用，为了减少人工添加任务的时间成本，也为了提供任务的复用性，本系统支持采用模板的方式进行批量下发任务，模板中定义好的任务的脚本以及顺序，仅需要任务选择测试终端即可实现批量下发，极大减少人工介入的时间成本。相关实现如下:

```
def model_create():
    model_names=db().select(db.model.model_name,groupby=db.model.model_name)
```

```
        names=['']
        cur_select=request.vars.value
        for name in model_names:
            names.append(name.model_name)
        form=FORM('选择模板:

',SELECT(names,_name='test',_id="model",value=cur_select,_onchange='cur_select=value;location.href="model
_create?value="+cur_select'), A('    没有模板?自定义一个',_href=URL("default","model_define")))
        model_tasks=db(db.model.model_name==cur_select).select(db.model.ALL)
        session.model_tasks=model_tasks
        ...获取模板任务下的计算机信息并展示到页面中
        return dict(form=form)
```

任务创建过程中需要经历多个页面，不同页面共用参数传递使用 web2py 自带的 session 全局参数进行传递，比如创建任务的具体信息。

任务展示实现方式等同脚本展示，此处省略。

C. 结果展示

因各种终端测试需要的结果多样性，无法对其进行统一格式，因此在实现时要求测试脚本输出的结果文件内容以 xml 格式，若需要对测试结果数据进行分析则进入数据分析模块进行深化的数据分析行为。结果展示相关实现如下:

控制器（controller）:

```
def result_detail():
    ...获取测试结果，过程省略
    detail=''
    detail=T("执行结果:").xml()
    detail+="<br />"
    detail+=(TEXTAREA(result_tmp[1]).xml())
    detail+=(T("执行 log:  ").xml())
    detail+="<br />"
    detail+=(TEXTAREA(log[1]).xml())
  return XML(detail)
```

视图（view）:

```
{{left_sidebar_enabled=False,('message' in globals())}}
{{extend 'layout.html'}}
{{block left_sidebar}}
...左侧展示列表菜单，过程省略
{{end}}
<div class="result-show" id='out'>...右侧具体结果展示，id 为 out
```

其中对于 view 中左侧菜单触发对应测试结果展示到右侧具体结果，通过 onclick 事件触发，调用控制器实现对应结果展示，如展示特定任务结果:

```
<a name= "result" onclick="ajax('{{=URL('default','result_detail', args=[result_list[i]['task_id']])}}',['letter-a'],'out');jQuery('textarea').readOnly=true;">任务结果</a>
```

2. 任务调度

整个系统的核心模块，主要实现从任务创建后下发执行到上传结果过程，以下分别从服务端到客户端角度解说实现方式。

A. 服务端

考虑任务调度服务端的易扩展性以及支持并发请求需求，系统采用 soaplib2.0 组件，通过 WSDL 接口对外提供任务下发、脚本下发、测试进度反馈以及结果上传功能。接口设计如:

```
class TestService(DefinitionBase):
    @soap(String,String,String,String, _returns=String)
    def heartbeat(self,name,ip,mac,obj):
        ...心跳处理，过程省略
  @soap(String,String, _returns=String)
    def get_task_list(self,computer_name,obj):
        ...接收任务，过程省略
    @soap(String, _returns=String)
    def get_shell_content(self,file_name):
            ...获取脚本，过程省略
    @soap(String,String,String,_returns=String)
```

```
def task_status(self,task_name,computer_name,obj):
    ...任务状态，过程省略
@soap(String,String,String,String,String,_returns=String)
def up_process(self,task_name,compute_name,status,log_file_id,out_file_id):
    ...更新任务执行状态，过程省略
@soap(String,String,String,_returns=String)
def post_file(self,table_name,file_name,file_content):
    ...上传文件，过程省略
```

WSDL 接口可实现自由扩展，接口实现完成之后，通过 wsgi 对外提供 WSDL 服务，关键实现过程如下:

```
from wsgiref.simple_server import make_server
soap_application = soaplib.core.Application([TestService], 'http://127.0.0.1:8989/')
 wsgi_application = wsgi.Application(soap_application)
 server = make_server('127.0.0.1',8989, wsgi_application)
 server.serve_forever()
```

B. 客户端

客户端任务调度在调用服务器提供的 WSDL 接口使用，可以通过 suds 组件提供的接口进行连接，以下以心跳包操作为例:

```
from suds.client import Client
test_client=Client(url,cache=None)
test_client.service.heartbeat(host_name,host_ip,OBJECT)
```

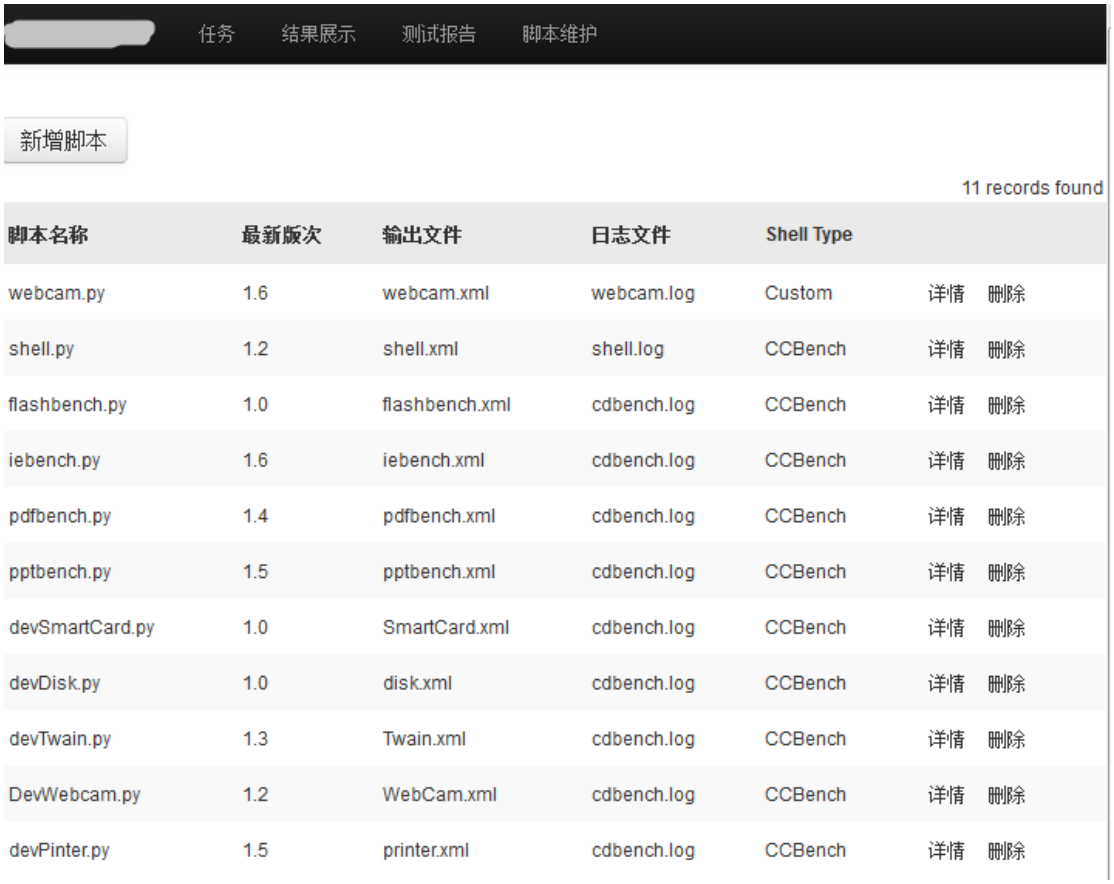
在客户端任务调度执行过程中考虑减少本身调度对测试结果的干扰，在调度机制上采用线性堵塞式调度,(线性指单线程,堵塞式指执行脚本使用堵塞式函数 os.popen(cmd))

3. 数据分析*

基于测试结果规格的多样性，为了满足保存需求，测试结果以文件 xml 格式存储数据库中，对于不同任务类型的结果数据如何处理需要在数据分析中进行，该过程根据实际需要进行自定义扩展，比如对 OA 办公体验进行数据分析行为。

四、测试使用

测试环境中，待测云终端采用 Centerm 和海思联姻产品 C10，结合测试的云系统为 XenDesktop 6.5 发布的 win7 系统，对其 OA 办公的体验进行测试。



脚本名称	最新版次	输出文件	日志文件	Shell Type	
webcam.py	1.6	webcam.xml	webcam.log	Custom	详情 删除
shell.py	1.2	shell.xml	shell.log	CCBench	详情 删除
flashbench.py	1.0	flashbench.xml	cdbench.log	CCBench	详情 删除
iebench.py	1.6	iebench.xml	cdbench.log	CCBench	详情 删除
pdfbench.py	1.4	pdfbench.xml	cdbench.log	CCBench	详情 删除
pptbench.py	1.5	pptbench.xml	cdbench.log	CCBench	详情 删除
devSmartCard.py	1.0	SmartCard.xml	cdbench.log	CCBench	详情 删除
devDisk.py	1.0	disk.xml	cdbench.log	CCBench	详情 删除
devTwain.py	1.3	Twain.xml	cdbench.log	CCBench	详情 删除
DevWebcam.py	1.2	WebCam.xml	cdbench.log	CCBench	详情 删除
devPinter.py	1.5	printer.xml	cdbench.log	CCBench	详情 删除

（a）脚本维护界面

任务名称	任务类型	任务脚本	执行时间	执行次数	执行间隔	说明	适用客户机	依赖任务	依赖客户机	
ccbench01	VM	iebench.py	2014-11-05 15:25:00	1	60	IE方面的测试	CCBench-01			删除
ccbench02	VM	pdfbench.py	2014-11-05 15:26:00	1	60	office中pdf的测试	CCBench-01			删除
ccbench03	VM	pptbench.py	2014-11-05 15:26:00	1	60	office中ppt的测试	CCBench-01			删除
ccbench04	VM	flashbench.py	2014-11-05 15:33:00	1	60	flash方面的测试	CCBench-01			删除

（b）任务下发界面

（c）任务调度执行

（d）任务结果展示 ■

参考文献

soaplib 官网：https://github.com/soaplib/soaplib

mysqldb 官网：http://sourceforge.net/projects/mysql-python/

suds 参考资料：https://fedorahosted.org/suds/wiki/Documentation

浅谈软件测试项目的风险管理

◆作者：田辉

风险管理往往被软件测试项目组所忽视，但随着 CNAS CL45-2013 的发布，风险管理将被第三方软件测评机构正式纳入管理体系。相对于软件开发项目的风险管理，软件测试项目的风险管理才刚刚起步。软件测试作为软件开发过程的一部分，软件测试项目的风险管理有其自身的特点。本文从第三方软件测评机构的立场出发，谈论了软件测试项目风险管理的过程及其活动。

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在的重要性。软件测试作为软件开发过程的一部分，也存在着风险。显而易见，软件测试项目的风险管理是项目风险管理的一种特殊形式。如果能开展风险管理，重视风险的评估，并制定积极的风险应对计划，就可以最大限度的避免风险或者降低风险所带来的损失。

2013 年发布的 CNAS CL45-2013 规定的“件测试项目管理括测试项目跟踪控制测试计划的实施进度和风险”，意味着风险管理将被第三方软件测评机构正式纳入管理体系。

纵观软件项目的风险管理，自 20 世纪 80 年代 Barry Boehm 比较详细地论述了软件开发中的风险，并提出了软件风险管理 Boehm 模型以来，已发展了 30 余年。相较于软件开发，第三方软件测试在中国才发展了二十余年，很多方面还有待规范。软件测试项目近几年才刚刚引入风险管理。如何将软件开发项目风险管理的经验与第三方软件测试相结合，如何做好第三方软件测试项目的风险管理是一个值得考虑的问题。

一、项目风险管理的术语

（1）风险 Risk

遭受损失的可能性。

（2）风险管理 Risk Management

一种问题分析的手段．它采用风险概率分析，对某情况的风险进行权衡研究，以便

更精确地了解所涉及的风险. 风险管理包括风险的识别、分析、优先级排序和控制。

（3）项目风险管理 Project Risk Management

是指通过风险识别、风险分析和风险评价去认识项目的风险，并以此为基础合理地使用各种风险应对措施、管理方法技术和手段，对项目的风险实行有效的控制，妥善的处理风险事件造成的不利后果，以最少的成本保证项目总体目标实现的管理工作。

（4）风险管理计划 risk management plan

描述一个项目将要执行风险管理活动的计划的集合。

（5）概率 probability

风险可能发生的程度。可以用等级来描述风险的概率，如难以置信、不太可能、可能性极小、偶然、很可能、频繁等。

（6）风险类别 Risk Category

对风险源类型的描述（如人、机、料、法、环、时间等）

（7）风险状态 Risk State

与单个风险有关的项目风险信息。

（8）风险阈值 Risk Threshold

印发一些共利益者采取措施的条件。

二、软件测试项目风险管理过程概要图

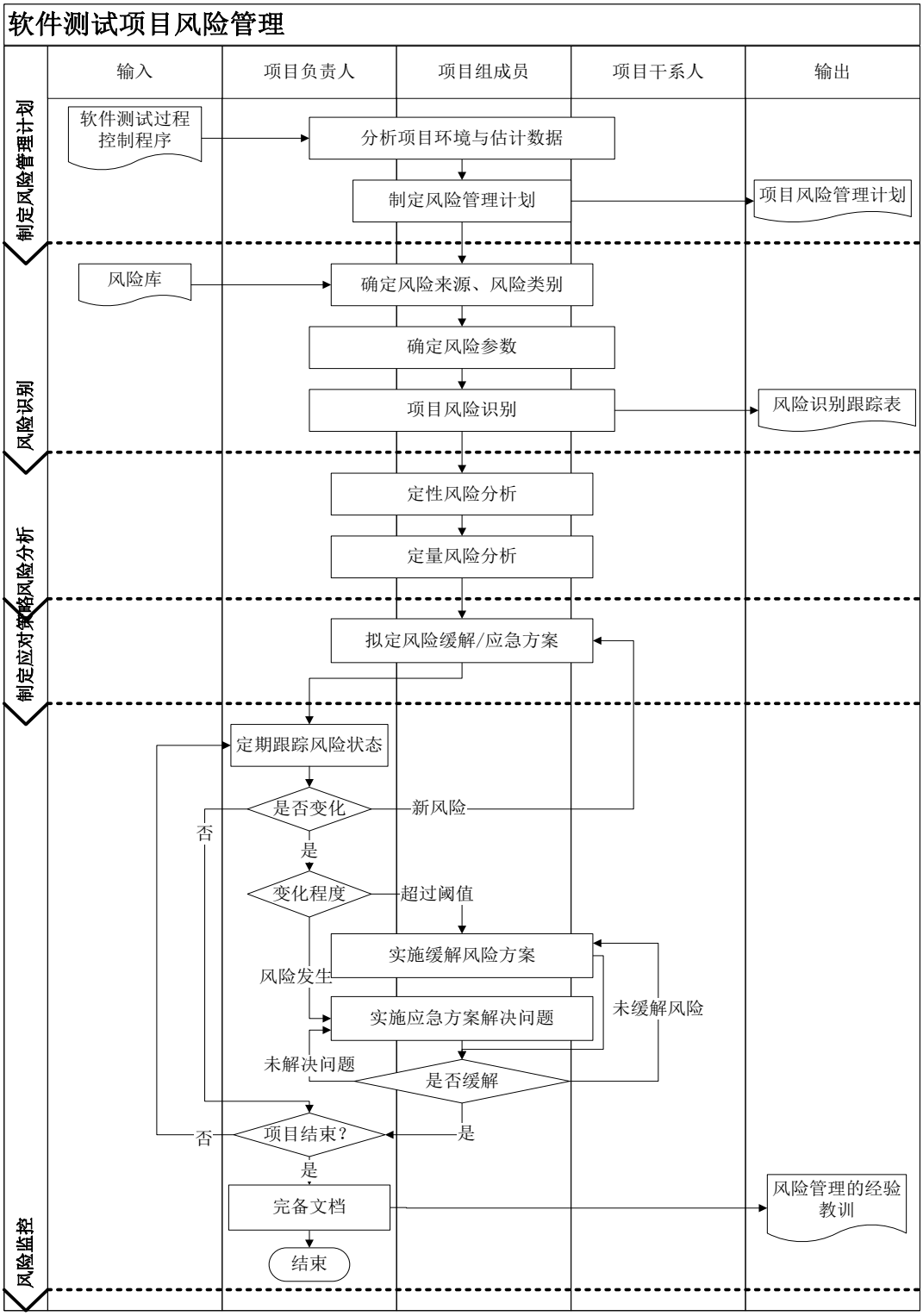


图 1 软件测试项目风险管理过程概要图

三、软件测试项目风险管理过程

1.制定风险管理计划

编制风险管理计划是用来确定项目风险管理相关的活动计划安排的工作，是项目风

险管理的首要工作。

作为项目的风险管理人员，项目负责人要有很好的洞察力、分析识别能力；良好的沟通能力，组织能力；对风险发生、发展、消亡等生存周期有很深刻的认识；对异常有很好的把握能力、处理能力；项目风险管理的经验。

在测试任务单下达、项目组成立后，项目负责人召集项目成员、项目干系人，依据测试过程的控制程序、已经签订的测试合同或测试委托书，以会议的方式分析项目可能存在的风险，依据工作安排，与项目成员讨论，一起编制风险管理计划。分析形成项目《风险管理计划》，须由项目负责人审核批准。

风险管理计划主要描述如何为项目处理和执行风险管理活动。风险管理计划中需要定义风险管理活动、风险级别、类型等内容。一份完整的风险管理计划要包含：风险管理方法，角色与职责，风险预算，风险管理的工具和技术、风险监控频度（制定时间表），风险类别，风险发生概率，风险影响，项目干系人对风险的容忍度，风险跟踪过程等。

作为第三方软件测评机构，开展的测试项目具有高度的相似性，历史项目的风险管理经验具有很高的可借鉴性，因此该类机构采用风险库的技术和方法来制定风险管理计划较为适合。利用风险库进行风险识别容易使项目负责人思路开阔、想到本项目会有哪些潜在的风险。

风险库是基于以前类似项目信息及其他相关信息编制的风险识别核对图表，还包括风险等级及处理标准的定义。风险库一般按照风险来源排列。

2.风险识别

对项目进行风险管理，首先必须对存在的风险进行识别，以明确对项目构成威胁的因素，便于制定规避风险和降低风险的计划和策略。在第三方软件测评机构中，风险识别从测试任务下达开始启动，至项目结束关闭。风险识别不是一次性的工作，贯穿于软件测试项目始终。信息的全面性、及时性和准确性决定了风险识别的质量和结果的可靠性和精确性。

风险识别的主要内容是识别和确定出项目究竟有哪些潜在的风险，这些项目风险究竟有哪些基本的特性，引起这些风险的主要因素是什么，这些项目风险可能会影响项目哪些方面。风险识别的结果是一份风险列表，其中记录了项目中所有识别出的风险。

进行风险识别时，一般由项目负责人组织项目成员或项目干系人召开会议，集思广益和深度挖掘，分析项目的实际情况，依据组织风险库的一些经验性文档，排除一些几乎从来没有发生的风险（或这部分风险也没有采取过措施的），识别出本项目的风险，确定风险的类别、来源，并填写在《风险识别跟踪表》中的“风险列表”中。

风险识别使用的主要工具与技术有文件审查、信息收集技术、检查表、假设分析、图解技术等。

（1）文件审查。文件审查主要是对测试方案、项目开发文档和产品文档等进行系统和结构性的审查。

（2）信息收集技术。项目风险识别中所采用的信息搜集常用的有德尔菲（Delphi）法、头脑风暴法、访谈法、SWOT（Strength/Weakness/Opportunity/Threat，优势/劣势/机会/威胁）分析法。

（3）检查表。检查表是项目管理中用来记录和整理数据的常用工具。用它进行风险识别时，将项目可能发生的许多潜在风险列于一张表上，供识别人员进行检查核对，以判别项目中是否存在表中所列或类似的风险。

（4）假设分析。每个项目都是根据一套假定、设想或者假设进行构思与制定的。假设分析是检验假设有效性的一种技术。它辨认不精确、不一致、不完整的假设对项目所造成的风险。

（5）图解技术。图解技术主要包括因果分析图、系统或过程的流程图、影像图等。

第三方软件测评机构一般从人、机、料、法、环等五个方面进行测试项目的管理。在识别测试项目隐藏的风险时，也从人、机、料、法、环等五个方面来全方位的分析和罗列项目级可能隐藏的风险。由于在软件测试中时间是一个非常重要的属性，所以也将时间纳入分析范围。下面列举软件测试项目中常见的风险。

（1）人（主要指测试人员）

业务不熟：测试人员对被测系统的业务流程不熟悉，体现在对需求的理解上把握不准、理解不透侧、理解错误等。

测试人员变动：离职，岗位调动，请假等。

定位效应：测试过的可靠的功能，特别是在多次回归且没有发现问题，在此后往往会认为此功能是可靠的。

疲态：某一些功能点一直由某一位测试人员测试，经过多次回归后，测试人员对该功能点的测试显示出倦意和缺乏兴趣。

同化效应：经过和开发的长时间接触，往往会被开发的思维逻辑所同化，渐渐丧失从用户角度出发的测试观察点。

人员能力不足：如某些项目存在技术难度，测试人员的测试能力和水平导致测试进展缓慢，项目延期。

沟通协调风险：如测试过程中涉及的角色较多，存在不同人员、角色之间的沟通、协作，存在误解、沟通不畅的情况。

（2）机（主要指测试所用设备）

测试所需设备无法按时到位。

设备故障。

（3）料（指测试相关文档）

文档缺失：如客户提供的项目文档不全。

需求风险：如对软件质量需求或产品的特性理解不准确，导致测试范围分析存在误差，遗漏部分需求或者执行了错误的测试方式；需求变更导致测试用例变更，同步时存在误差。

测试用例设计不到位：忽视了一些边界条件、深层次的逻辑、用户场景，用例没有完全覆盖需求等。

测试数据不充分：如未准备足够的测试数据。

质量标准不统一：如可靠性的测试。

代码质量风险：软件代码质量差，导致缺陷较多，容易出现测试的遗漏。

（4）法（即测试方法和实施）

错误或缺失测试方法：没有采用正确的测试方法，或某些测试方法被忽视，如边界测试、异常处理等，导致测试不充分。

场景的缺失或部分缺失：例如忽视了业务场景的测试。

测试用例实施不充分：测试用例没有得到百分之百的执行，如有些测试用例被有意或无意的遗漏。

回归测试有选择性的执行测试用例。

缺陷风险：某些缺陷偶发，难以重现，容易被遗漏。

（5）环（即测试环境）

样品的被测软件版本与合同规定的软件版本不一致。

搭建的测试环境与要求的测试环境不一致：如 CPU 频率，内存大小等。

搭建的测试环境与实际运行环境不完全一致，造成测试结果的误差。

（6）时（即测试时间）

测试时间不足：如项目实际规模比估算规模大很多，项目时间不足，但没有调整测试计划等。

测试时间延长：如前期质量保证活动执行不到位，导致后期的返工工作量过大等。

3.风险分析

风险分析分为定性风险分析和定量风险分析两个步骤。

（1）定性风险分析

定性风险分析是指对已识别风险的可能性及影响大小的评估过程，该过程按风险对项目目标潜在影响的轻重缓急进行优先级排序，并为定量风险分析奠定基础。

项目负责人组织项目成员或项目干系人对已经识别的风险进行定性分析，确定严重性、可能性等参数，确定该项目的风险等级及排列顺序。项目负责人将结果记录在《风险识别跟踪表》中。

由于第三方软件测试具有重复性、可监督性和动态性的特点，适用于软件测试项目的定性风险分析工具和技术主要为风险概率和影响矩阵、风险数据质量分析、风险分类、风险紧迫性评估。

①风险概率和影响矩阵

风险的概率及影响是针对具体风险事件而言，而不是整个项目。风险概率与风险影响可以用极高、高、中、低、极低等定性术语加以描述。风险概率和影响矩阵即风险级别评定矩阵，可以将概率与影响的标度结合起来，以此为依据建立一个对风险或风险情况评定等级（极低、低、中、高、甚高）的矩阵。高概率与高影响风险可能需要作进一步分析，包括量化以及积极的风险管理。进行风险级别评定时，每项风险要有自己的矩阵与风险标度。

②风险数据质量分析

风险数据质量分析就是评估有关风险的数据对风险管理的有用程度的一种技术，它包括检查人们对风险的理解程度，以及风险数据的精确性、质量、可靠性和完整性。

（2）定量风险分析

定性风险分析过程完成后，可进入定量风险分析过程或直接进入风险应对规划过程。定量风险分析是指对定性风险分析过程中作为项目需求存在的重大影响而排序在先的风险进行分析，并就风险分配一个数值。软件测试项目的定量风险分析与一般项目的定量风险分析一样，是在不确定情况下进行决策的一种量化方法，常采用 EMV、PERT、蒙特卡罗分析等技术。

①EMV

EMV（Expected Money Value，期望货币值），用以计算在将来某种情况发生或不发生情况下的平均结果。机会的期望货币价值一般表示为正数，而风险的期望货币价值一般被表示为负数。每个可能结果的数值与其发生概率相乘之后加总，即得出期望货币价值。

每种情况的损益期望值为:

$$EMV = \sum_{i=1}^{m} P_i X_i$$

其中，Pi 是情况 i 发生的概率，Xi 为 i 情况下风险的期望货币价值。

②PERT

PERT（Plan Evaluation and Review Technique，计划评审技术）是利用网络分析制定计划以及对计划予以评价的技术。

③蒙特卡罗分析

蒙特卡罗分析法又称统计实验法，是随机地从每个不确定性因素中抽取样本，对整个项目进行一次计算，重复进行很多次，模拟各式各样的不确定性组合，获得各种组合下的很多个结果。通过统计和处理这些结果数据，找出项目变化的规律。

4.制定风险应对策略

风险分析完成后，项目负责人组织项目成员，也可以在必要的情况下组织项目干系人，依据已经识别的风险及等级排序，制定相应的应对策略。项目负责人将制定的应对策略记录在《风险识别跟踪表》中。

由于风险的影响分为积极影响和消极影响两种，所以针对不同影响采取不同的应对策略。存在消极影响的风险采取回避、转移与减轻的策略，存在积极影响的风险采取开拓、分享或提高的策略。

（1）回避消极风险策略是指当项目风险潜在威胁的可能性极大，并会带来严重的后果，无法转移又不能承受时，通过改变项目来规避风险。

（2）转移消极风险策略是在项目中使用最频繁的做法是通过合作伙伴、项目外包与担保等手段将项目风险转移到第三方。

（3）减轻消极风险策略是通过缓和或预知等手段来减轻风险，降低风险发生的可能性或减少风险发生后后果的影响程度和范围，设法把不利的风险事件的概率或后果降低到一个可接受的临界值。

（4）开拓积极风险策略是通过确保机会肯定实现而消除与特定积极风险相关的不确定性。

（5）分享积极风险策略是指将风险的责任分配给最能为项目的利益获取机会的第三方，包括建立风险分享合作关系。

（6）提高积极风险策略旨在通过提高积极风险的概率或其积极影响，识别并最大程度地发挥这些积极风险的驱动因素，致力于改变机会的“大小”。

5.风险跟踪

项目负责人定期跟踪、监控风险的现状。只要项目没有结束，项目负责人依据项目风险管理计划定期或事件驱动地跟踪、监控风险的现状。定期分析当前风险的可能性、

严重性等参数是否发生变化。定期识别是否有新的可能的风险会产生。并根据跟踪监控情况更新《风险识别跟踪表》。

（1）已识别风险的处理

当《风险识别跟踪表》中风险的参数系数已经超过了规定的阈值时，项目负责人组织项目成员，依据已经建立的风险应对策略，执行缓解措施。没有超过规定的阈值的风险可以不作任何处理。

同时项目负责人应组织制定一套完整的“风险应急预案”，用来处理风险转化为问题时的处理。项当风险状态发生变化时，及时组织人员对这个风险进行分析、按照制定的缓解措施方案执行。

①当风险参数变化超过规定的阈值，但没有转化为异常，或没有造成对项目有明显的影响时，项目负责人组织人员依据已经制定的缓解方案，对风险进行缓解。

②当风险参数变化超过规定的阈值，并已经对项目有明显的影响，产生了不良后果或产生灾难性后果，也就是说风险发生，转化为异常时，项目负责人应及时组织项目成员、项目总监、项目的干系人分析这个异常，制定一套良好的应急方案。异常有可能导致项目失败。项目负责人组织人员执行这个应急方案来解决这个异常，直到各相关项目干系人认为措施已经缓解了风险为止。

③项目负责人应将风险跟踪活动中的，分析分析、制定方案、缓解风险等过程、结果，花费的时间、人力、资源等都记录在《风险识别跟踪表》中。

（2）突发风险的处理

当风险发生时，及时组织人员对这个风险进行分析、制定缓解措施方案，并按照制定的缓解措施方案执行，并通报项目总监或部门经理。

①当风险参数变化超过规定的阈值，但没有转化为异常，或没有造成对项目有明显的影响时，项目负责人组织人员依据已经制定的缓解方案，对风险进行缓解。

②当风险参数变化超过规定的阈值，并已经对项目有明显的影响，产生了不良后果或产生灾难性后果，也就是说风险发生，转化为异常时，项目负责人应及时组织项目成员、项目总监、项目的干系人分析这个异常，制定一套良好的应急方案。异常有可能导致项目失败。项目负责人组织人员执行这个应急方案来解决这个异常，直到各相关项目

干系人认为措施已经缓解了风险为止。

③项目负责人应将风险跟踪活动中的，分析分析、制定方案、缓解风险等过程、结果，花费的时间、人力、资源等一一记录在《风险识别跟踪表》中。

6.总结经验

当项目结束时，项目负责人总结风险管理过程中的一些建议、经验教训、新发现的可能的风险等，形成项目的总的建议、经验教训，可以将这部分写入项目总结报告，为所在组织的过程财富做贡献。

四、相关记录

《风险管理计划》

《风险识别跟踪表》 ■

参考文献

《GB/T 20918-2007 信息技术 软件生存周期过程 风险管理》

《GBT 11457-2006 信息技术 软件工程术语》

《软件测试风险分析》

《测试风险的管理》

《软件测试常见风险分析》

软件测试自我修养(一):修心三问

◆作者：刘铭辉

一直以来想把"软件测试人员的自我修养"作为一个系列，连贯的表达出来。一方面目的是平时工作之余，适当的总结精炼知识以便后续工作中回顾；另一方面是和大家分享，共同进步。

“授人以鱼，不如授之以渔” 说的是传授给人知识，不如传授给人学习知识的方法。今天我想针对于此从思维层面再做一个升华：“授之以渔，则先令人悟之”。

做好软件测试，首先具备的修养是需要弄明白三个问题。这就是上面讲到要的“悟”。假如开发人员修改提交了 BUG，我们使用“三问”的思想进行测试，对于测试人员了解需求会起到很大的帮助。

何以悟“三问“。您一定会问：“哪三问？”

举个例子，咱们来设计一个场景：想象一下，您走在马路上，迎面走来一个陌生人二话没说扇了你一巴掌。

这时出于理性思考您会想到什么。“我无意中得罪他了？”、“这人是不是精神上有问题？”、“他认错人了？”等等。无论您想到什么，终究归结到一个问题，那就是“打我的原因是什么”。

第一问：“原因是什么”

开发人员提交被修复的缺陷，测试人员在展开测试工作之前需要弄明白的第一个问题“此缺陷的原因是什么”。以便于针对问题的原因进行测试分析、测试计划、开展测试工作。事物之间的联系是多样的，在软件测试的思维中引入“先行后续”、“引起与被引起”的关系，加之唯物辩证方法论。可以令我们更为透彻的理解基于此缺陷提出的原因，从而进一步深入的理解业务层面的需求。

通过沟通、分析弄明白了：走在街上被扇耳光是因为长得太丑。有人会有疑问：“虽说长得太丑，也不能扇耳光吧。为什么要扇耳光呢”。这就是我们要弄明白的第二

个问题“缺陷为什么要这样做”

第二问：根源--“为何要这样修复”

缺陷被修复的方案可以有多个，基于业务、技术实现、设计风险等等角度考虑是开发人员否选择了最优方案。了解“为何要这样修复”有助与我们设计测试案例，并综合衡量测试的风险，便于使问题暴露，增大对相应风险系数较大的测试点进行关注度。

亚里士多德说过,一切存在物都由本源构成，一切存在物最初都从其中产生，最后又复归为它。弄明白“为何要这样做”能使我们在进行软件测试工作的时候最大限度的追溯其本源，制定测试计划。使缺陷更大程度的暴露。

弄明白“扇耳光原因是丑的无可救药了”。不过话说回来，你可以用语言来感化我，为什么要打我呢，为何不用语言来感化我、为何不去打别人呢。

从软件测试的角度，我们要弄明白的第三个问题“为什么不这样做”

第三问：启发--“为何不那样做”

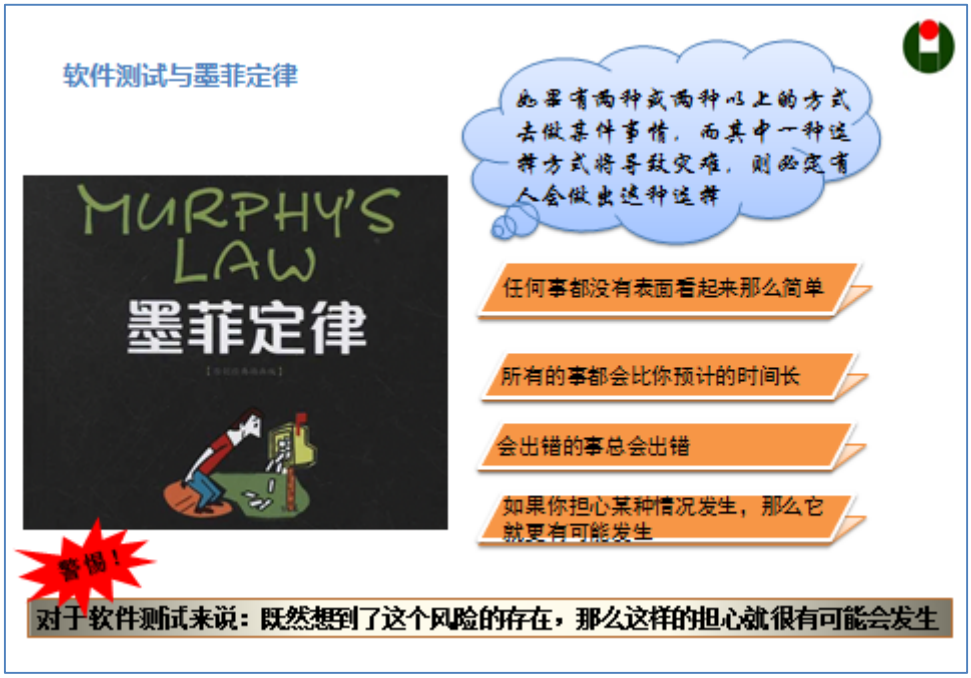
作为一个专业的软件测试人员，只了解问题的原因、以及根源为何要这样修复是不够的。从思维发散的角度来想，需要我们更进一步的考虑“another way”。这一点很重要。这个缺陷的修复为何用 PLAN A，为何不用 PLAN B？除了 PLAN B 外，还有没有 PLAN C？弄明白这些问题，便于我们更清楚的熟悉业务、理解系统、提升测试技能、设计测试案例。这些需求背后的需求是诸多软件测试人员成长的瓶颈。

鄙人不才，望“三问”作为软件测试的方法论能使大家修测试之心，悟测试之性。工作之余多沟通交流，一起成长进步。■

软件测试与墨菲定律

◆ 作者：刘铭辉

提起墨菲定律，大家一定会想起最近在各大院线热映的《星际穿越》影片，男主角库伯深爱的女儿墨菲。影片中墨菲定律似乎贯穿着整部电影：比如库伯送给女儿墨菲手表时，墨菲担心父亲是否能如期而归，而结果却等了 100 多年……再比如库伯希望尽早离开 1 小时 7 年的星球担心自己回到地球时女儿长得和自己一样大、甚至比自己老，但当他回去的时候，结果却是见到了临终前 100 多岁的女儿……



墨菲定律的原话是这样的：If there are two or more ways to do something, and one of those ways can result in a catastrophe, then someone will do it.（如果有两种或两种以上的方式去做某件事情，而其中一种选择方式将导致灾难，则必定有人会做出这种选择）

对于软件测试来说，我想分两个层面来理解墨菲定律与软件测试的关系

1. 开发者层面

举例说明，开发人员小明准备针对某个 BUG 进行程序修复。假设修复这个 BUG 有三种设计方案，Plan A、Plan B、Plan C，且 Plan B 会造成系统崩溃的风险，则根据墨菲定律，必定存在某种概率，小明会选择 Plan B。

这时你可能会有疑问，是什么因素导致小明去选择 Plan B？对于小明来讲，影响到自己去选择造成系统崩溃的方案是多样化的：比如自身对程序理解的误区、编写代码不良的习惯、程序代码的边界欠缺考虑、系统的弱点的敏感度低等等。软件测试人员可以根据以上这些因子，结合小明的性格、习惯、心理素质等权重，简单的利用科学评价方法预估出其选择 PlanB 的概率，并及时抛出风险。

2. 测试者层面

同样例举，测试人员小刚在功能测试需求分析的时候，针对系统可能发生的问题进行了详细的记录。但是在编写功能测试案例的时候，认为繁杂的案例加重了自己的工作量，就凭借着经验去掉了一些认为不重要测试案例，导致生产问题发生。

不仅是小刚，相信这样的心理每位测试人员都有过。因此，在未经过详细、分析评审的基础上，擅自做“执行”or“不执行”测试案例选择的时候，这种风险就已经产生了。

根据“墨菲定律”：

1）任何事都没有表面看起来那么简单；

2）所有的事都会比你预计的时间长；

3）会出错的事总会出错；

4）如果你担心某种情况发生，那么它就更有可能发生。

总体来说，生产安全无小事。当我们担心因为某个原因可能会导致风险，那么这样的担心就很有可能会发生。由此可知，我们意识到风险发生的存在某些概率时，请及时抛出风险，以便项目团队积极应对。■

单元测试之打桩

◆作者：白开水

单元测试（unit testing），是指对软件中的最小可测试单元进行检查和验证。对于单元测试中单元的含义，一般来说，要根据实际情况去判定其具体含义，它可以是一个函数，单个类。总的来说，单元就是人为规定的最小的被测功能模块。

单元测试可以归纳为：异常测试（也称为白盒测试、压力测试、可靠性测试），用来确认代码的结构可靠性，能处理所有可行的输入以及输入组合，不会产生预料之外的异常。功能测试，用于验证所建立模块是否符合需求和功能是否正常工作。在单元级别创建功能测试涉及到人工输入，以指定特定的输入和状态条件、以及预期的输出。

单元测试通常涉及手工编写测试集、指定输入数据以及为缺少的函数提供桩函数。而对于桩函数我们又该如何去构建呢？

首先，对于项目中缺少的函数需要构造桩函数，它有可能是低层的一个接口函数，SDK、MFC 等提供给用户使用的 API 函数。在对这些 API 函数构建桩函数的时候，我们可以做到尽量的简洁，在测试的过程中我们并不需要对这些函数的内部进行测试，只需要它为我们测试的代码提供一个返回值，或者只是为了让我们所测试的代码更好的执行下面的流程。

简单的介绍个 MFC 的例子:

```
void CLightSettingDlg::OnPaint()
{
if (IsIconic())
{
CPaintDC dc(this); // device context for painting
        RePaint();
}
else
```

```
{
    CDialog::OnPaint();
}
    }
```

在这里输入测试用例后执行的都是 else 下的语句，为了能够执行到 IF 判断为 TRUE 的语句。需要对 IsIconic()打桩，手动返回 TRUE 的值。

桩函数如下:

```
#include <afxwin.h>
::BOOL (::CWnd::Test_Stub_IsIconic)(void) const
{
  Return true;
}
```

对上面的一段代码再次进行测试，当测试用例输入后，运行测试在 CPaintDC dc(this); 遇到异常；这是因为 CPaintDC dc(this); 在当前窗口创建 *CPaintDC* 对象；它是由 MFC 提供的一个构造函数，这个语句是需要初始化 this 这个对象。初始化一个图形化的组件，它会涉及到初始化整个应用程序，包括 GUI 的组件等一系列额外的工作。然后在测试的过程也会涉及到打开窗口，按按钮等的交互式方式，而这些就不是单元测试的范畴。这时候就会对 CPaintDC dc(this);进行打桩，使程序继续运行。

为了在执行测试的过程中，让后续的代码能够被测试到，引入了桩函数，以上的例子不需要我们设计较复杂的桩函数，便可以执行到。在平常的测试中，对于一些外部的函数，我们也可以构造简单的桩函数返回特定值，或者初始化某些的变量。让当初执行不到（无需测试）的代码正常通过，提高测试的覆盖率。

参考文献:

《白盒测试之道》北京航空航天大学出版社

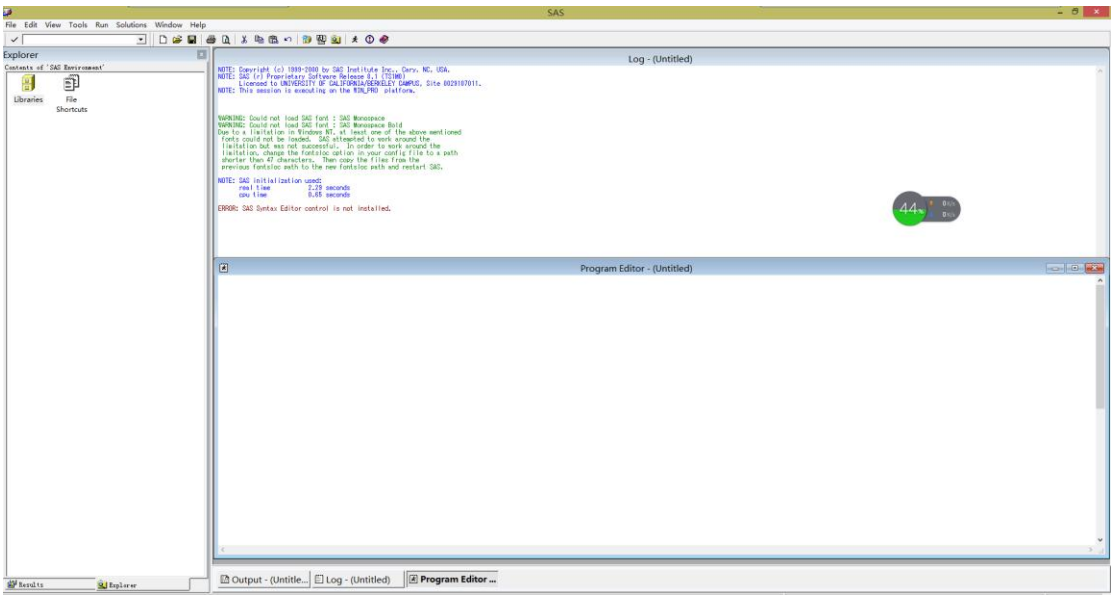
SAS 软件的使用和统计学分析的初步介绍

◆ 作者：吴晨昊

一般而言我们都会使用 Excel 来统计测试结果，除了 Excel 之外，还有 SAS 等软件，也是可以统计测试结果的，本人也是 SAS 的初学者，现在我就给大家介绍一下 SAS 的简单使用，随着我不断的学习统计学的知识，我也希望今后能更深入的探究这些统计学软件的功能，并将这些功能和测试相关联。

第一部分：SAS 软件的基本使用

1: 打开 SAS 软件。

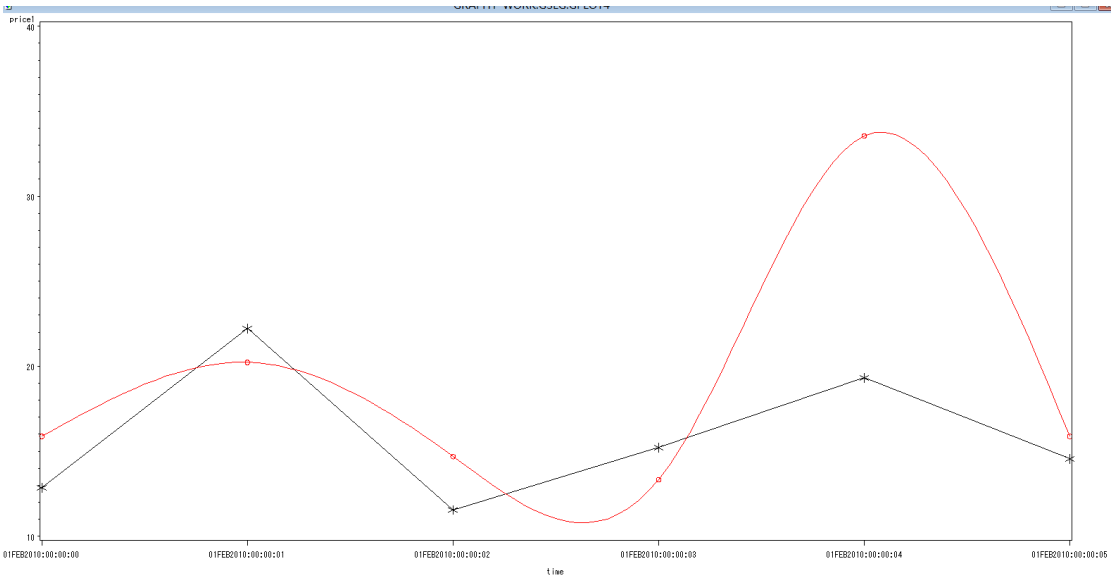


2: 在“Program Editor”输入框中输入如下代码(初始时间是 2010 年 2 月 1 日 00:00:00，每一秒钟取一个数据点，有两行统计数据，并以两种不同的形式显示出来，一种是直线，一种是曲线。)

```
data ex;
```

```
input price1 price2;
time=intnx('second', '01FEB2010:00:00:00'dt,_n_-1);
format time datetime20.;
cards;
12.85 15.88
22.22 20.23
11.55 14.69
15.21 13.33
19.33 33.55
14.56 15.88
;
proc print data=ex;
proc gplot data=ex;
plot price1*time=1 price2*time=2/overlay;
symbol1 c=black v=star i=join;
symbol2 c=red v=circle i=spline;
run;
```

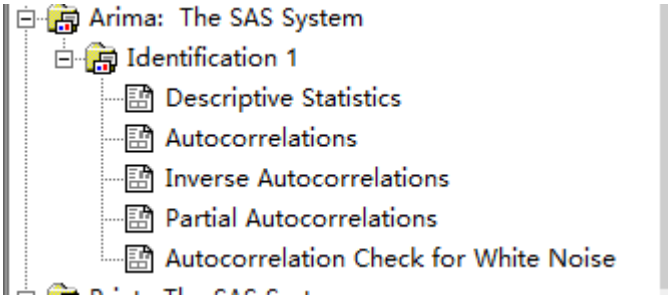
3：点击上方导航栏中的”Submit”按钮，我们就可以看到运行的结果：



4: 在“Program Editor”输入框中输入如下代码,则可以查看到这些数据的平均值和标准偏差

```
data ex;
input price@@;
time=intnx('second', '01FEB2010:00:00:00'dt,_n_-1);
format time datetime20.;
cards;
235 258 301 144 189 134 200 236 316 236 412 362 214 233 255 344 199 280 300 312 321 412 523 124 325;
proc arima data=ex;
identify var=price;
proc print data=ex;
run;
```

双击左侧 Result 中的 Arima:The SAS System->Identification 1->Descriptive Statistics



我们可以看到，这些数据的均值是 274.6， 标准偏差是 91.61135，总的数据个数是 25。

The ARIMA Procedure

Name of Variable = price

Mean of Working Series	274.6
Standard Deviation	91.61135
Number of Observations	25

如果我们双击“Autocorrelation Check for White Noise”

Autocorrelation Check for White Noise

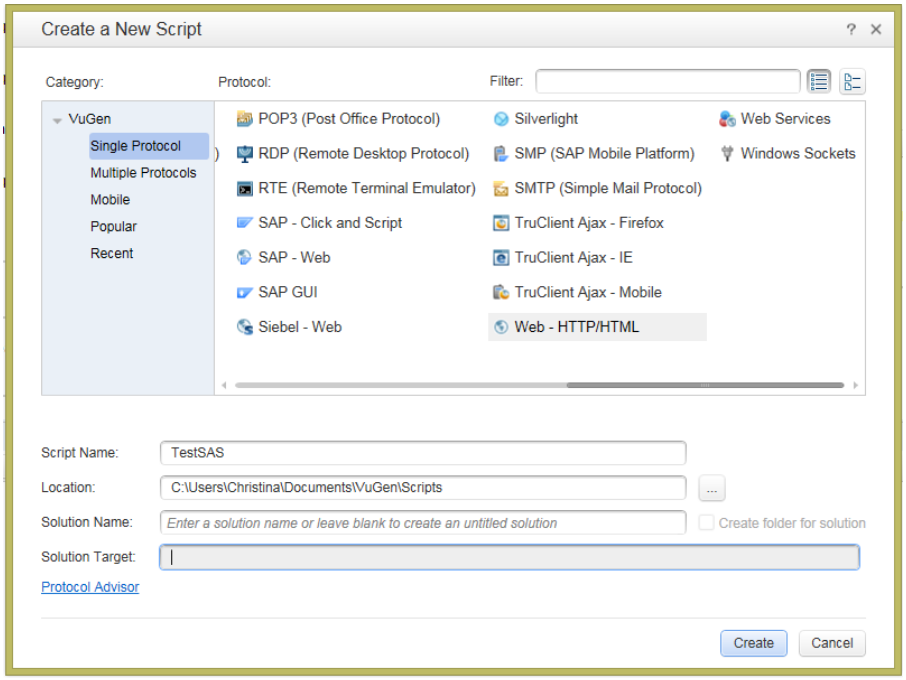
To Lag	Chi-Square	DF	Pr > ChiSq	Autocorrelations					
6	2.66	6	0.8503	0.125	0.123	0.081	0.030	-0.132	-0.164

显示出 QLB 统计量服从 χ^2 分布，自由度为 6，QLB 统计量的 P 值为 0.8503 所以它是白噪声，基本上并没有什么规律。

第二部分： LoadRunner 的数据生成

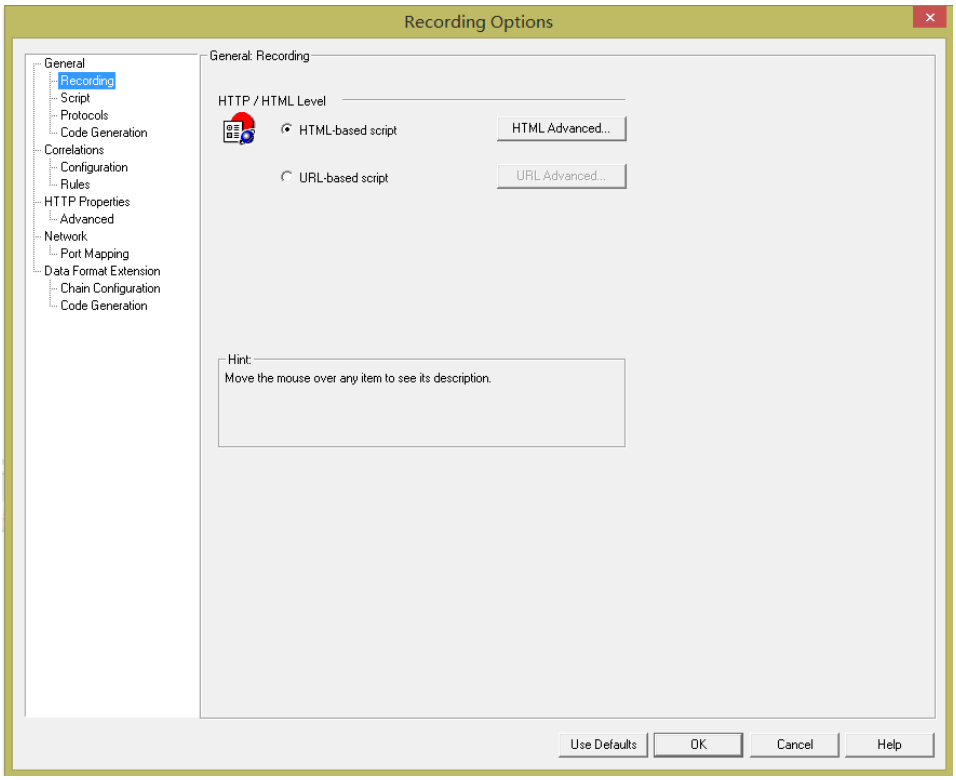
1: 双击 “Virtual User Generator”图标，打开 VUG。

2: 点击“File->new script and solution”-选择 Web-HTTP/HTML,然后点击“Create”。

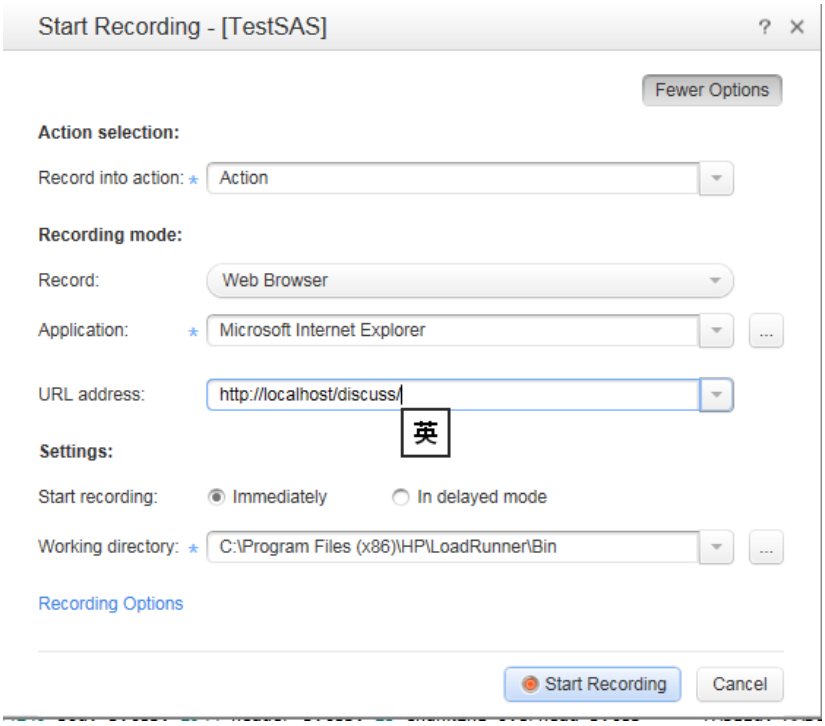


3: 选择录制模式：

点击“Record”-> “Recording Options”,选择“HTML based script”。



4: 点击“Record”按钮，选择要录制的 URL，然后点击“Start Recording”按钮。

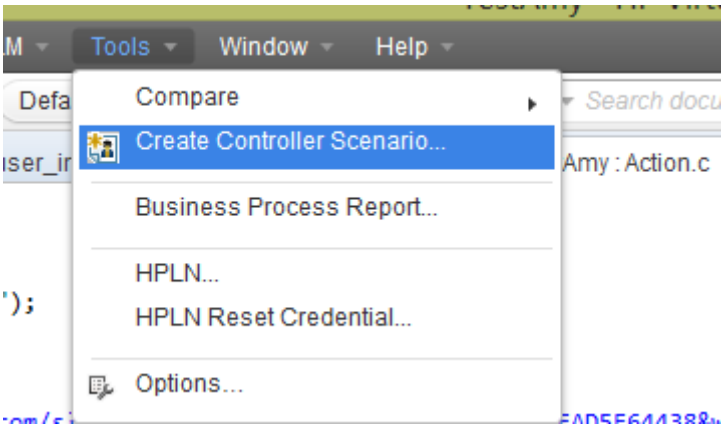


5: 我们可以看到，系统会弹出一个正在 Recording 的控件，同时，IE 浏览器也显示了，我们所设置的 URL 地址。

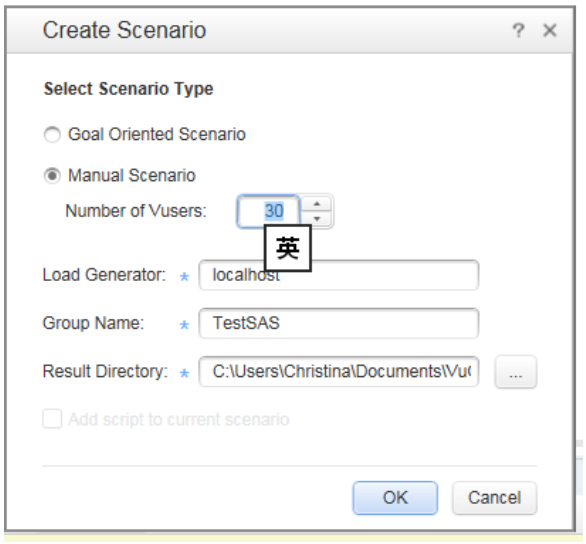
6: 点击网页中的链接，录制完成脚本之后，设置 Transaction。

```
Action()
{

    web_url("discuss",
        "URL=http://localhost/discuss/",
        "Resource=0",
        "RecContentType=text/html",
        "Referer=",
        "Snapshot=t2.inf",
        "Mode=HTML",
        LAST);

    web_set_sockets_option("SSL_VERSION", "TLS1.1");

    lr_think_time(6);

    lr_start_transaction("start");


    web_url("view.asp",
        "URL=http://localhost/discuss/view.asp?ContId=1",
        "Resource=0",
        "RecContentType=text/html",
        "Referer=http://localhost/discuss/",
        "Snapshot=t4.inf",
        "Mode=HTML",
        LAST);

    lr_think_time(4);

    web_link("返回首页",
        "Text=返回首页",
        "Snapshot=t5.inf",
        LAST);

    return 0;
    lr_end_transaction("end", LR_AUTO);

}
```

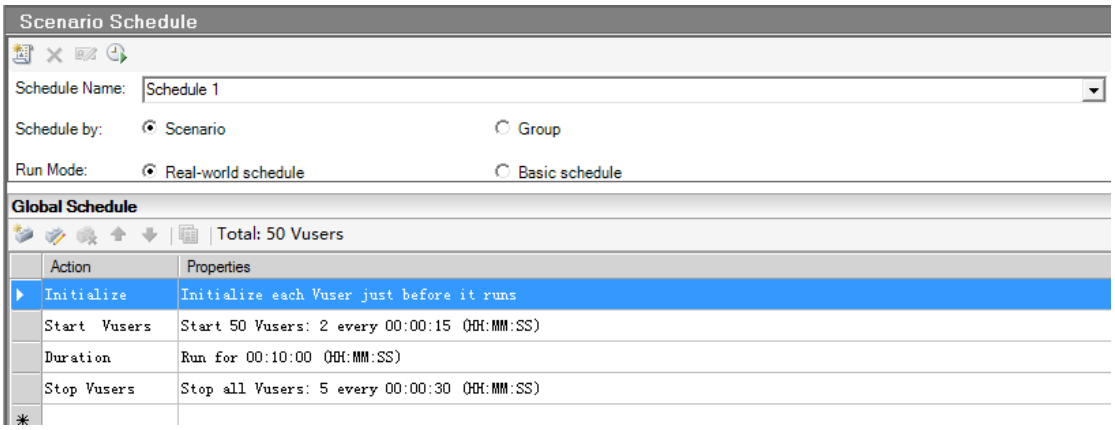
7: 点击“Tool-> Create Controller Scenario”。



8: 弹出如下的对话框，并点击“OK”按钮，弹出 Controller 的界面。



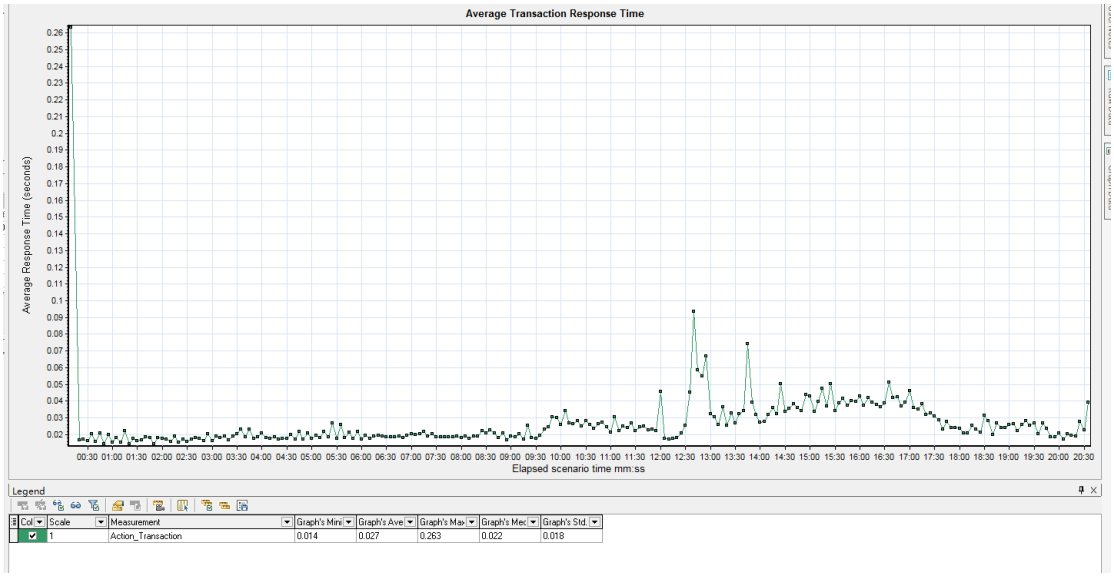
设置 50 个用户的压力之下，在全部加载完毕之后，连续运行 10 分钟，系统的运行情况。



9: 开始运行。

第三部分: SAS 和统计学的分析

1: 运行完毕，查看 Analysis 中的图表。



显示的是 Average Transaction Response Time

在右侧，点击“Graph Data”-> Export Graph Data to XLS file-> 取文件名为 GraphData.XLS，保存在桌面。

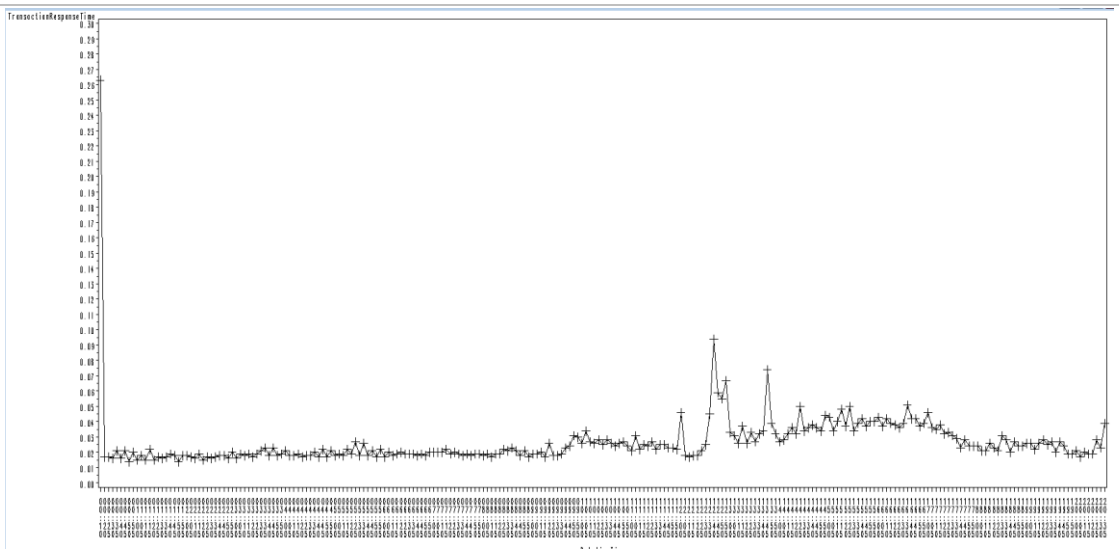
于是就生成了一个 Excel 文档，除了第一条和第二条数据之间间隔为 10 秒之外，其余间隔都是 5 秒。

	A	B	C
1	Relative Time	Action_Transaction	
2	0:10	0.263	
3	0:20	0.017	
4	0:25	0.017	
5	0:30	0.016	
6	0:35	0.021	
7	0:40	0.016	
8	0:45	0.021	
9	0:50	0.014	
10	0:55	0.02	
11	1:00	0.015	
12	1:05	0.018	
13	1:10	0.015	
14	1:15	0.022	
15	1:20	0.015	
16	1:25	0.017	
17	1:30	0.016	
18	1:35	0.017	
19	1:40	0.019	
20	1:45	0.018	
21	1:50	0.014	
22	1:55	0.018	
23	2:00	0.018	
24	2:05	0.017	
25	2:10	0.016	
26	2:15	0.019	
27	2:20	0.015	
28	2:25	0.017	
29	2:30	0.016	
30	2:35	0.017	
31	2:40	0.018	
32	2:45	0.018	
33	2:50	0.016	
34	2:55	0.02	
35	3:00	0.016	
36	3:05	0.019	
37	3:10	0.018	
38	3:15	0.019	
39	3:20	0.017	
40	3:25	0.019	
41	3:30	0.021	

同时，我们可以看到，图标下方的统计信息

Color	Scale	Measurement	Graph's Minimum	Graph's Average	Graph's Maximum	Graph's Median	Graph's Std. Deviation
TRUE	1	Action_Transaction	0.014	0.027	0.263	0.022	0.018

2: 我们将这些数据放在 SAS 中，看是否会出来一致的结果:



使用的语句如下

```
data ratio;
input    RelativeTime$ Action_Transaction@;
cards;
00:10  0.263
00:20  0.017
00:25  0.017
00:30  0.016
00:35  0.021
00:40  0.016
00:45  0.021
00:50  0.014
00:55  0.02
01:00  0.015
01:05  0.018
01:10  0.015
01:15  0.022
01:20  0.015
01:25  0.017
……
此处省略过多数据
```

… …

01:30 0.016

01:35 0.017

01:40 0.019

01:45 0.018

01:50 0.014

01:55 0.018

02:00 0.018

02:05 0.017

02:10 0.016

02:15 0.019

02:20 0.015

02:25 0.017

19:15 0.026

19:20 0.028

19:25 0.025

19:30 0.027

19:35 0.02

19:40 0.027

19:45 0.024

19:50 0.019

19:55 0.019

20:00 0.021

20:05 0.017

20:10 0.02

20:15 0.019

20:20 0.019

20:25 0.028

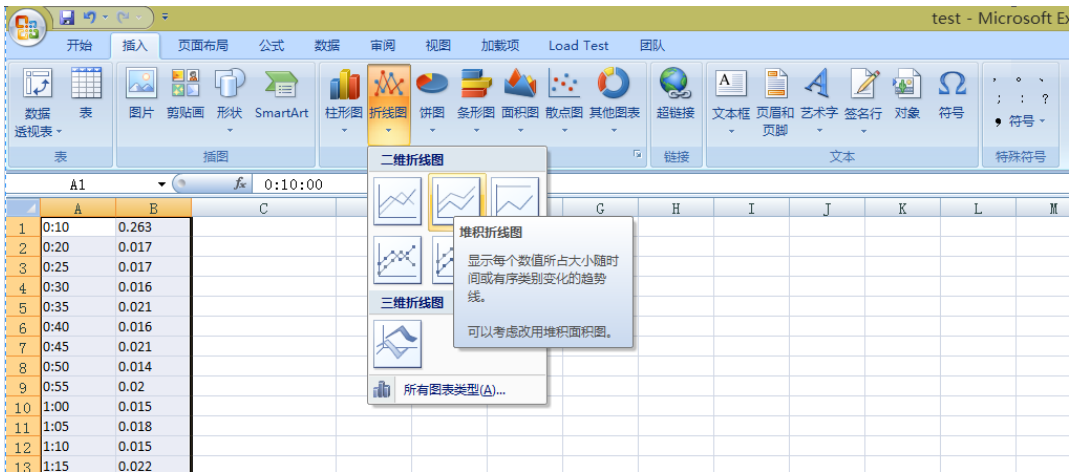
20:30 0.023

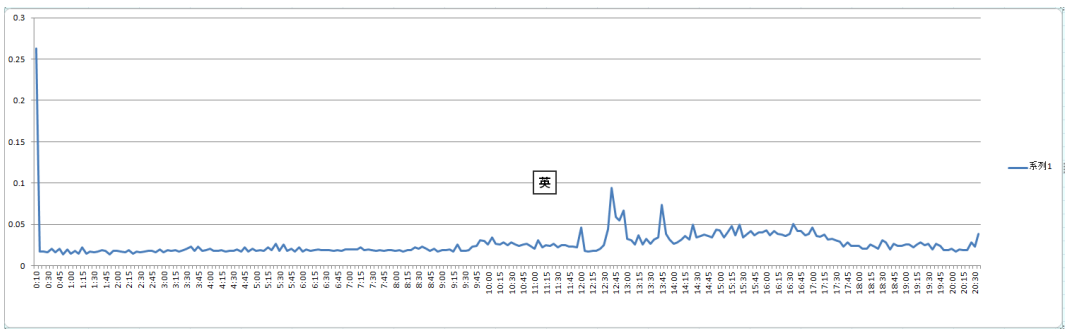
20:35 0.039

```
;
goptions reset=all ;
proc arima data=ratio;
identify var=Action_Transaction;
proc print data=ratio;
axis2 label=('TransactionResponseTime')
        order=(0 to 0.3 by 0.01);
proc gplot;
plot Action_Transaction*RelativeTime=1/   vaxis=axis2;
symbol1 v=plus i=join c=black h=2;
run;
```

使用 Excel 也能画出这样的图:

在 Excel 中，选中两列数值，选择插入->折线图->堆积折线图，就可以生成折线图了。但是它的横坐标的数值数量会随着你放大或者缩小图表而增多或者减少。





3: SAS 中可以查看到这些数据的平均值和标准偏差。

The ARIMA Procedure

Name of Variable = Action_Transaction

Mean of Working Series	0.026665
Standard Deviation	0.018454
Number of Observations	245

4: 此时的标准偏差相当于 excel 中的 STDEVP(B1:B245)函数，其总体选用的是 n-1。

问 1: 如果要求，方差小于 1，那么请问这些数据是否可以接受(显著水平为 0.05)?

因为 χ^2 的统计量为: $\frac{\sum_{i=1}^{n}(x_i-\bar{x})^2}{\sigma_0^2}$ =0.018454*0.018454*244/1=0.083

在 Excel 中，如果统计量的值小于 CHIINV(0.05,244)=281.44，那么就是可以接受的。

因为 0.083 小于 281.44 所以可以说是可以接受的。

SAS 中的 χ^2 分布 CHIINV(0.05,244)写法是:

```
data _null_;
test=CINV(0.95,244);
put test;
run;
```

我的解释:

以上统计的方法可以用来测试压力测试:

压力测试就是测试一个系统在什么级别的压力之下能完好无损的长时间运行，或者在符合一定的条件之下运行。如果系统在某个级别的压力之下，运行失常，那么就说明这次的压力测试是不达标的。

在这个级别的运行一次完毕之后，就可以进行统计，并得出运行结论。

问 2: 如果我对这个网站进行 2 次运行，得到 2 批数据，可以验证这 2 批数据的方

差是否一致(显著水平为 0.05)?

Excel 表述法:

右边界: FINV(0.025,244,244)

左边界: 1/ FINV(0.025,244,244)

统计量是 $\frac{S_1^2}{S_2^2}$

如果这个统计量的数值是在左边界和右边界之间，那么就认为这两次数据的方差是一致的，没有什么区别。

在 SAS 上设置 F 分布 FINV(0.025,244,244)

```
data _null_;
test=FINV(0.975,244,244);
put test;
run;
```

我的解释:

以上统计的方法可以用来测试负载测试:

负载测试就是测试一个系统在运行时当给予它的压强有多大的时候，它会崩溃，或者它会不符合要求。如果在某次增加负载之后，系统仍然能够照常运行，那么说明此次的负载测试是达标的。

我可以这么测试: 10 个用户运行 5 分钟，算出它们的 S_1^2；然后 20 个用户运行 5 分钟，算出它们的 S_2^2； 然后计算统计量 $\frac{S_1^2}{S_2^2}$ 是否满足 F 分布的统计方法。如果满足，可以再进行 30 个用户运行 5 分钟的数值观察，一直持续下去，直到它不再服从 F 分布的统计方法为止。因此，此方法可以用来分析某个负载情况之下，系统运行的响应时间的稳定程度的差异。

最后，欢迎大家提出自己的宝贵意见和建议，谢谢。

参考资料: 王燕.应用时间序列分析(第二版).中国人民大学出版社.2012 年 6 月第 8 次印刷.

Selenium 教程：31 +最好的 Selenium 免费培训课程

◆ 译者：飞燕儿

很多读者提出了许多关于 Selenium 的问题，那么今天我们就开启免费的 Selenium 基础教程。在这个基础教程系列中，我们会覆盖所有的 Selenium 概念，我们会举一些简单的例子让大家更好的理解 Selenium Package 的使用。

这些 Selenium 教程对初学者向 Selenium 高级测试发展非常有用。都是从非常基础的 Selenium 概念教程开始，然后一步步向比较高级的课程开展 如怎样创建框架，设计 Selenium 架构和精华商业架构 BDD。

注：我们会尽快发布更新的课程内容。请不要错过任何教程内容。 请做好标签，以及时学习我们 Selenium 课程新内容。

那么怎么开始学习 Selenium 呢？

结合 Selenium 免费课程开始自学 Selenium 是最好的。先看教程，然后在家练习实例，最后在 comment 中提出对课程存在的问题，我们会通过邮件来解答你的疑问。

Selenium 专家们也可以在 comment 中帮助解答读者的问题。

这是我们能尽的最大努力来帮助大家学习和掌握目前最流行的软件测试工具！

Selenium 简介:

很高兴我们再次推出了软件测试的培训教程另一系列课程。本教程的宗旨就是让你们成为最流行工具 Selenium 的测试自动化专家。

这个教程中我们会从 Selenium 不同方面进行考虑。Selenium 不仅仅是一个工具；它是许多单个工具的集群。我们会详细讲解其中的一些工具，并会提供一些比较实用的实例。

在看比较有趣、实用的章节之前，我们先看看 Selenium 为我们提供了哪些功能。

为什么用 Selenium?

行业趋势发展表明，现在测试都朝着自动化测试发展。重复的手工测试场景给测试工程师们提出了新的需求，那就是自动化测试代替手工测试。

执行自动化测试的好处有许多，主要存在于以下几点:

- 支持测试用例重复执行;
- 帮助完成大型项目的测试矩阵覆盖;
- 可以展开平行测试
- 鼓励无人值守执行测试
- 提高准确性，从而减少人为错误
- 节约时间成本和金钱成本

这些好处带来的结果:

- 高投资回报率
- 更快速地投入市场

自动化测试好处，也能很容易理解，大家可以在软件测试领域展开讨论。

另一个最普遍问到的问题就是----

- 最适合我的自动化测试工具是什么
- 需要花什么成本吗
- 工具方便使用吗

一个最合适的回答解决上面几个问题就是基于 Web 的自动化测试工具 Selenium。理由:

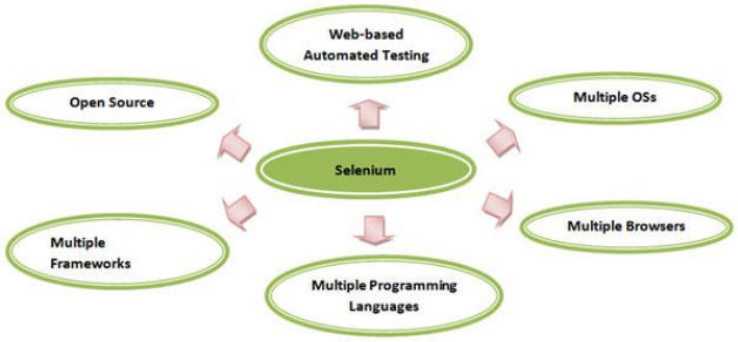
- 开源工具
- 有大量用户使用，还有帮助论坛
- 兼容多个浏览器和平台

- 灵活的开发库
- 支持多种语言

扫一眼 Selenium

Selenium 是最流行的自动化测试工具之一。它是支持和解决基于 Web 程序功能自动化测试的工具，其支持绝大多数的浏览器和平台。由于它是开源工具，所以 Selenium 逐渐成为测试领域使用最多的工具。

Selenium 支持一系列的浏览器，技术和平台：



Selenium supports a broad range of browsers, technologies and platforms

Selenium 组件

Selenium 不仅仅是一个测试工具，还包含了一系列的测试工具包，同时 Selenium 也是一个组件套。Selenium 这些组件或工具包就是为解决不同的测试和测试环境需求的。

这个组件包是由以下几个工具组成：

- Selenium 集成开发环境（IDE）
- Selenium 远程控制（RC）
- Selenium WebDriver
- Selenium Grid

Selenium RC 和 Webdriver 结合使用就是我们熟知的 Selenium2. Selenium RC 单独使用就是 Selenium1.

Selenium 工具简介

Selenium core

Selenium 是 Thoughtworks 公司叫 Jason Huggins 的工程师持续努力的结果。 由于 Jason 的工作是负责程序内部测试以及对时间和成本的控制，意识到用自动化测试工具代替手工来保证测试的质量和准确性是非常必要的。

于是在 2004 年初他开发了一款叫 ‘JavaScriptTestRunner’ 的 JavaScrip 工具，这工具能够像人操作浏览器那样自动控制浏览器。

从此以后，Jason 开始向大众演示讲解这款工具。甚至花钱来讨论这款工具应该归到哪类开源工具里面，并且还为其他 Web 应用开发可重用的测试框架。

这款工具就是我们现在熟知的 ‘Selenium Core’

Selenium IDE（Selenium 集成开发环境）

Selenium IDE 是由 Shinya Kasatani 开发的。当研究 Selenium Core 时，他意识到这个 JavaScript 代码能扩展创建一个集成开发环境（IDE），这个插件可以集成到 Mozilla Firefox 中。把 IDE 集成到 Firefox 浏览器中就可以录制和回放用户做的操作。 2006 年 Selenium IDE 成为 Selenium 包的一部分。 这个工具对社区来说具有巨大的价值和潜力。

Selenium IDE 是 Selenium 包中最简单的工具。它的记录和回放功能对掌握任何一门编程语言的测试来说能够很容易掌握。Selenium IDE 有几个优点，也是因为这些特点导致 SeleniumIDE 不适合用在更高级的测试脚本语言中。

Selenium IDE 优点和缺点:

优点:

1. 易录制和回放
2. 可将测试脚本转换成 Html，Java，C#和其他不同的语言
3. 没有严格要求测试人员掌握某一门开发语言
4. 使用文件日志插件记录日志
5. 调试能设置断点
6. 灵活性&可扩展性

缺点:

1. 只支持 Firefox 浏览器
2. 不支持迭代和条件语句
3. 不支持错误处理
4. 不支持测试脚本独立或群
5. 不支持数据库测试

IDE 缺点实际上不是 Selenium 的缺点，而仅仅是 IDE 的一些局限性。这些局限性可以通过使用 Selenium RC 或者 WebDriver 来解决。

Selenium RC（Selenium 远程控制）

Selenium RC 是一款用 Java 编写的可供用户使用 Web 程序之前选择的编程语言来构造测试脚本。

Selenium RC 的出现是为了解决使用 Selenium IDE 或 Core 所存在的缺点。

Selenium Core 存在的漏洞和局限性使用户非常难发挥工具的优点。所以 Selenium Core 会在测试流程中较难进行，任务较难完成。

一个至关重要的约束就是同源策略（Same origin policy）。

Same origin policy（SOP）存在的问题:

SOP 的问题就是当我们试图访问的文档源与原文档有差别时，会拒绝用户访问原文档的 DOM。

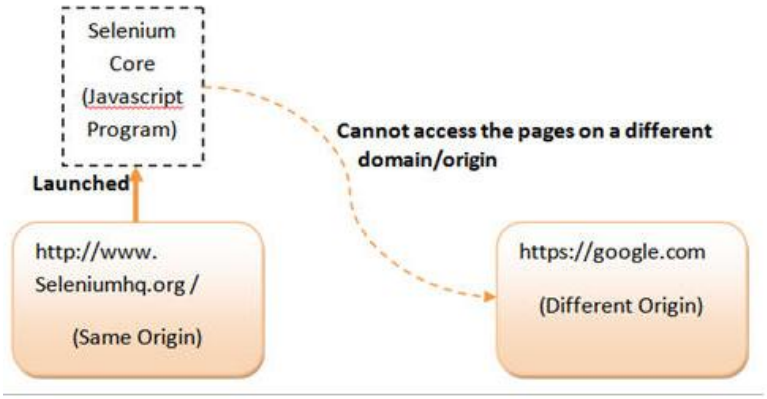
Origin 是 scheme，主机和 URL 端口顺序的组合。例如，URL

http://www.seleniumhq.org/projects/ ，origin 就是 HTTP 协议，Seleniumqh.org,端口 80 对应的集合。

因此 Selenium Core（JavaScrip 程序）不能访问与 origin 不同的的元素。

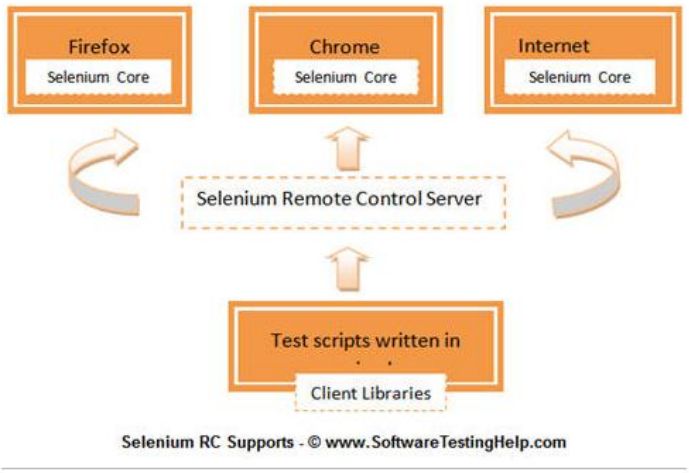
如，如果我已经从 http://www.seleniumhq.org/ 运行了 Javascript 程序，然后我能访问相同域中的页面 如“http://www.seleniumhq.org/projects/” 或 http://www.seleniumhq.org/download/。其他域像 Google.com,yahoo.com 就不能访问。

所以，要使用 Selenium Core 测试程序，必须要在 Selenium Core 中安装整个程序及 Web 服务器来解决 SOP 问题。



所以在使用 Selenium Core 时为了控制 SOP 而不创建应用程序副本，就需要使用 Selenium RC。当 Jason Huggins 演示 Selenium 时，另外一个在 ThoughtWorks 叫 Paul Hammant 的同事建议 SOP 的工作区和工具可以结合我们所选的编程语言。所以最后 Selenium RC 就面世了。

不像 Selenium IDE，Selenium RC 支持大量的浏览器和平台。



工作流程描述

1. 用户使用期望的编程语言创建测试脚本
2. 对每种编程语言而言，这是指定的客户端库
3. Selenium 服务器将测试命令编译后转换成 JavaScript 命令，并发送到浏览器。
4. 浏览器使用 Selenium Core 执行命令，并将结果发送回 Selenium 服务器。
5. Selenium 服务传送测试结果到客户端库。

在创建 Selenium RC 脚本前必须要做的一些准备:

1. 编程语言 –Java，C#，Python 等。
2. 集成开发环境-Eclipse，Netbeans 等。
3. 测试框架（可选）--JUnit，TestNGA 等。
4. 学习 selenium RC 课程。

Selenium RC 优缺点:

来看看 Selenium RC 有哪些优缺点，参考如下的数据。

优点:

1. 支持编程语言和结构
2. 支持一些浏览器和平台
3. 支持用户自定义工具的创建，如用于定制框架的术语/异常。

4. 支持错误处理和数据库测试
5. 支持测试数据驱动测试
6. 支持日志和截图
7. 支持测试框架 如 TestNG 和 Junit

缺点:

1. 测试脚本与浏览器不直接接触，需要启动 Selenium RC 服务器来支持测试脚本执行。
2. 用户受 Selenium RC 编程知识束缚
3. 不能高效处理警告和指示
4. 不支持基于 WAP 程序的测试（Iphone、android）
5. 比 Selenium IDE 快但比 WebDriver 慢
6. 不支持监测器执行
7. 不能有效处理 Ajax 调用

Selenium Grid

有 Selenium RC 支持，测试生活让人变得积极，让人喜欢，直到集成趋势提出新的需求 在多平台和不同浏览器上执行相同或不同的测试脚本 以实现分布式测试执行，在不同环境的测试 并有效地保存执行时间。因此，为了完成这些新需求，Selenium grid 就被引用进来。

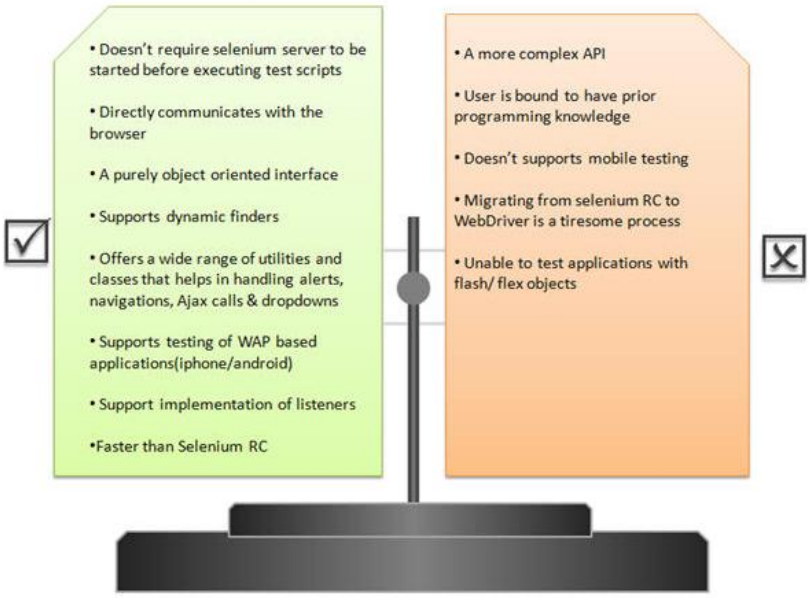
Selenium Grid 是由 Pat Lightbody 推出，为了解决需要在多个平台上同时执行测试套件。

Selenium Webdriver

Selenium WebDriver 是由 ThoughtWorks 另一位工程师 Simon Stewart 在 2006 年创建的。 WebDriver 也同样是一款与 Selenium RC 有微小差别的基于 Web 测试的工具。由于此工具是在 为每个 web 浏览器单个创建的独立客户端 基础上开发的；没有 Javascript 负载要求。这就导致在 Selenium RC 和 WebDriver 间的兼容分析。所以一款更强大的自动化测试工具 Selenium2 就开发出来了。

WebDriver 是一个干净的纯基于对象的框架。它利用浏览器自然的兼容而不用任何外围实体来实现自动化。随着需求增加，Webdriver 越来越流行，也拥有越来越多的用户群。

Selenium WebDriver 优缺点:



Advantages and disadvantages of Selenium WebDriver - © www.SoftwareTestingHelp.com

Selenium 3

Selenium 3 是比 Selenium2 更新的版本。它主要用于移动类和 web 类自动化测试。Selenium3 支持基于移动设备的 APP 测试，意思是说 WebDriver API 已扩展到工具解决移动类 APP 测试的需要。Selenium3 很快面世。

环境和技术

随着 Selenium 套件新工具的增加，环境和技术也越来越兼容。下面是 Selenium 系列工具所支持的环境和技术。

Browser Name	Selenium IDE	Selenium RC	WebDriver
Mozilla Firefox	Yes	Yes	Yes
Google Chrome	No	Yes	Yes
Internet Explorer	No	Yes	Yes
Opera	No	Yes	Yes
Safari	No	Yes	Yes
htmlUnit	No	No	Yes
Others	No	Partial Support Possible	Partial/Full Support Possible

Selenium Supported Browsers - © www.SoftwareTestingHelp.com

Supported Programming Languages

Programming Language	Selenium IDE	Selenium RC	WebDriver
Java	No (Can generate code)	Yes	Yes
C#	No (Can generate code)	Yes	Yes
PHP	No (Can generate code)	Yes	Yes
Perl	No (Can generate code)	Yes	Yes
Ruby	No (Can generate code)	Yes	Yes
Python	No (Can generate code)	Yes	Yes

Selenium Supported Languages - © www.SoftwareTestingHelp.com

Supported Operating Systems

Operating System	Selenium IDE	Selenium RC	WebDriver
Windows	Yes	Yes	Yes
Mac OS	Yes	Yes	Yes
Linux	Yes	Yes	Yes
Solaris	Yes	Yes	Yes

Selenium Supported OSes - © www.SoftwareTestingHelp.com

Supported Testing Frameworks

Testing Frameworks	Selenium IDE	Selenium RC	WebDriver
JUnit	No	Yes	Yes
NUnit	No	Yes	Yes
RSpec	No	Yes	Yes
TestNG	No	Yes	Yes
Unittest	No	Yes	Yes
Robot Framework SeleniumLibrary	No	Yes	Yes

Selenium Supported Frameworks - © www.SoftwareTestingHelp.com

总结:

这篇教程中，我们通过讲解 Selenium 不同组件，他们的使用方法及相互对比来让大家熟悉 Selenium 的这些套件。

下面是本教程学习重点:

Selenium 是 自动化测试工具的套件，每个自动化工具各有所长，弥补测试所需。

所有这些工具都从相同开源类形成的结构，并只支持基于 Web 的测试。

Selenium 套件由 4 个基本的组件组成：Selenium IDE，Selenium RC，Webdriver，Selenium Grid。

用户根据自己需要可以有更加选择 Selenium 更合适的工具。

Selenium IDE 是作为 Firefox 插件，更容易安装和使用。不要求用户提前掌握编程知识。Selenium IDE 对初次使用的用户来说是首选工具。

Selenium RC 是服务器，允许用户使用自己想用的编程语言创建测试脚本。它同样也支持脚本在大量的浏览器中执行。

Selenium Grid 跟 Selenium RC 相比有一个额外的功能：可以将测试脚本分布在不同的平台和浏览器同时执行，所以执行主动架构。

Webdriver 是一个完全不同的工具，比 Selenium RC 有不同的特点。 Selenium RC 和 WebDriver 的融合就是所熟知的 Selenium2。 Webdiver 直接和 web 浏览器交互，并利用兼容性进行自动化。

Selenium 3 是最期望加入的 Selenium 套件，很快将面世。Selenium3 更强大地支持移动类测试。

在下一个教程中，我们会讨论 Selenium IDE 的基本知识，SeleniumIDE 安装和它的特点。我们也会关注 Selenium IDE 的基本术语和专用术语。

下一个 Selenium 教程：Selenium IDE 简介和它的安装及 Selenium IDE 所有特性的详细研究。

为读者标记：当我们 Selenium 培训系列的下一个教程正在准备的时候，你们也可以查看 Selenium 官网来拓展关于 Selenium 套件及其工具。

作者简介

Shruti Shirivastava(这个教程的主要作者)，Amaresh Dhal，和 Pallavi Sharma 帮助我们为读者带来了这个培训。Shruti 目前是有着 4 年多自动化测试经验的资深测试工程师。她是 ISTQB 认证的专家，同时也是一个活跃的博客使用人，对解决测试相关的问题非常感兴趣 Amaresh 有 5 年多的手动和自动化经验，是 Webdriver，Grid 和框架方面专家。

自动化测试的一些随想

◆ 作者：郝强

1. 保证自动化测试成功的三块基石

保证软件项目的自动化测试工作成功，如果有三个方面能够做好，那么项目成功即自动化测试成功指日可待。那么，它们是什么呢？它们分别是数据、自动化，以及工具。在此我们分别来讨论一下它们。

数据：对于验证日益复杂的软件功能来讲，测试数据的有效性是极其重要的，并且如果测试数据是能够重复的，那么才能够保证自动化测试的效率。所以对于数据来讲，我们要求它是能够被定制的，能够恢复的并且可以重复使用的。如果上述这些内容能够是自动的，那么数据部分就应该准备的比较充分了。通常情况下，我们的手工测试数据与自动化测试数据是混杂在一起的，通常情况下会互相受影响。特别是当你的软件项目有许多个 release 的时候，你每次为之前的项目跟自动化测试脚本的时候你就会发现数据几乎不能使用了，我们差不多需要为每个脚本重复制作数据，这其中的成本是很大的，而且自动化测试有效率的特性也无法显现。当然对于这种情况有多种不同的解决方案，我的建议是自动化测试部分的数据是能够独立存在的。当然你得保证它的有效性，这里不做具体的讨论，大家可以尝试多思考交流解决。

自动化：关于自动化，我们主要是讲自动化测试，也就是说，我们的用户需求，能够被转化为自动化测试脚本，这些自动化测试脚本能够对软件进行测试，从而保证软件功能变便以及重构后，其功能及业务流程未被破坏。对于自动化测试来讲，我们希望是能够有较高的自动化测试覆盖率，从而获得大量时间成本。同时我们也希望较少的脚本的维护。所以针对于自动化测试脚本来讲，我们更应该从软件开发的角度去考虑，构建以及维护它。而不应该简单将手工测试用例转化为自动化测试。如果只是将手工测试用例转为自动化测试用例，我们将会花费大量的时间在维护测试脚本，测试对象库等等方

面。当然，我们这里提到自动化，我们也指软件工程的其它方面也被自动化，不只限于自动化测试。如果我们的整个工程大部分或绝大部分是能够自动化的，那么我们的生产力就会得到提升。例如：如果我们能够有自动化测试脚本，那么我们的自动化测试执行也应该是被自动调用的，并且最终的测试结果以及度量结果都应该是可以被自动获取的。简单来说，一切都应该被自动化。

工具：关于工具，我们更多也是关注在自动化测试工具。像常用的 QTP,Selenium 等等。对于企业来讲，工具的范围还是比较广的，不仅仅关注在自动化工具上。其实，如果单独来讲自动化方面，企业应该有自己的自动化测试工具的标准以及规范。即对于 web 应用我们使用何种工具，如何使用它等等。对于移动应用，我们又使用哪些工具，以及针对移动应用的使用规范。这些工具可以是商用的，也可以是开源的，当然也可以是我们自己开发的。对于工具来讲，我们的工具应该基于整个企业能够快速交付的整体框架之上，而且方方面面是有机结合在一起的，并非绝对独立存在的。即我们可能有整合数据的工具，也有用于测试的工具，也有用于度量的工具。这些工具是必须是可以随时集成到一起，可以互相通信，以保证我们在需要使用它们的时候，能够有最简便有效的获取最终我们要取得的数据，但绝不能允许每个部分都有自己独立的标准以及规范。

2. 自动化测试生产力

最近，被同事问到一个有趣的问题，就是自动化测试生产力。我们的自动化测试生产能力是多少，如何知道我们每年的自动化测试生产力是提升了，还是下降了。针对此总题，我觉得这其实是对要求我们能够有数据指标来衡量一个企业或团队的自动化测试能力。关于这个指标如何定义，我相信各个企业一定有自己的标准。但我想数据的获取应该是类似的，也就是说，那些参与运算的数据应该是差不太多的。在提及这些数据之前我想说一下关于手工测试能力的考察方法，通常大家会把每年写了多少测试用例，执行了多少测试，通过了多少，也有无效的 defect 有多少等等这些数据考虑进去。对于自动化来讲，我们也会考虑相关数据，如我们的自动化测试覆盖率，我们的脚本被运行的多少次，pass 多少次，我们节省了多少时间，我的脚本维护成本是多少等等，有了这些基础数据，我们就可以定制企业自己的自动化测试生产力模型，完成对自己企业或是团队的自动化测试能力的衡量工作。

3. 自动化测试框架

关于自动化测试框架，这其实是一个非常有意思的现象。为什么这么说呢，因为我

们会发现，大多数自动化测试团队都会实现一个自动化测试框架，然后做为标准或规范在团队内推广。而且一但有了自动化测试框架后，我们就会发现，我们的自动化测试脚本开发效率提高了，而且了也规范了许多。这是一件好事，但在这里我想说的事，不要过度框架化，自动化测试的框架还是要尽可能的遵循复用的原则。千万不要搞我之前曾经实现过的类似的关键字框架，我认为目前所有的所谓的关键字测试框架都是效率极其低下的，维护成本是极其高昂的。为什么呢，我们会有维护不完的关键字，我们需要对关键字进行长期大量的维护，当你的自动化测试是一个长期的项目的时候，你才会发现关键字驱动的框架的弱点。如果我们需要花费大量的维护工作在维护关键字上，我们的自动化可能存在问题。所以我这里并不推荐它。我的建议就是要像软件开发那样去考虑你的自动化测试。自动化测试专家一定是一个懂测试的开发人员或是懂开发的测试人员。你一定要在测试及开发领域工作多年，才有可能真正懂得为什么我们要将重心移至从开发角度去考虑自动化测试。可能我并不没有说明白，最终的目的是我们的自动化测试必须是高效的，少维护的，所以如果你使用的自动化测试方法，不能够实现这个需求的时候，那么它就是不应该被推广的。