

[www.qconferences.com](http://www.qconferences.com)

[www.qconbeijing.com](http://www.qconbeijing.com)



QCon北京2014大会 4月17—19日

伦敦 | 北京 | 东京 | 纽约 | 圣保罗 | 上海 | 旧金山

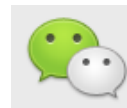
London · Beijing · Tokyo · New York · Sao Paulo · Shanghai · San Francisco

# QCon全球软件开发大会

International Software Development Conference



@InfoQ



infoqchina

软件  
正在改变世界!

# 特别感谢 QCon上海合作伙伴



# 顺势而为的Web自动化测试

东软集团

殷坤

2013年11月1日

Neusoft



## ➤ 技术发展对自动化测试的挑战

➤ 提高用户界面测试的敏捷程度

➤ 测试用例的高复用与自动装配

➤ 云计算助力测试环境高效管理

# Web应用开发常用技术

Web应用开发从架构设计、开发到部署目前都有非常成熟的技术

架构设计

组件化



编码实现

RIA框架



应用部署

云计算



开发过程

敏捷



Neusoft

# 对自动化测试应该引发的思考

技术的发展使开发效率逐步提高、交付周期逐渐缩短，而我们的自动化测试呢？

## 测试用例 稳定性

- 页面布局或样式发生调整后，测试用例还能运行么？
- UI框架更换或升级版本后，测试用例还能运行么？
- 浏览器更换或升级版本后，测试用例还能运行么？

## 测试用例 复用度

- 测试脚本按照什么粒度划分既便于复用又便于组织？
- 测试数据如何管理才能实现与测试脚本的灵活搭配？
- 如何根据组件装配变化自动组织可执行的测试用例？

## 测试环境 管理效率

- 是否可以不安装相关工具，就能够快速运行测试用例？
- 如何快速搭建各种测试环境，并提高资源使用效率？
- 如何快速把环境恢复到历史状态，以便重现特定问题？





➤技术发展对自动化测试的挑战

➤提高用户界面测试的敏捷程度

➤测试用例的高复用与自动装配

➤云计算助力测试环境高效管理

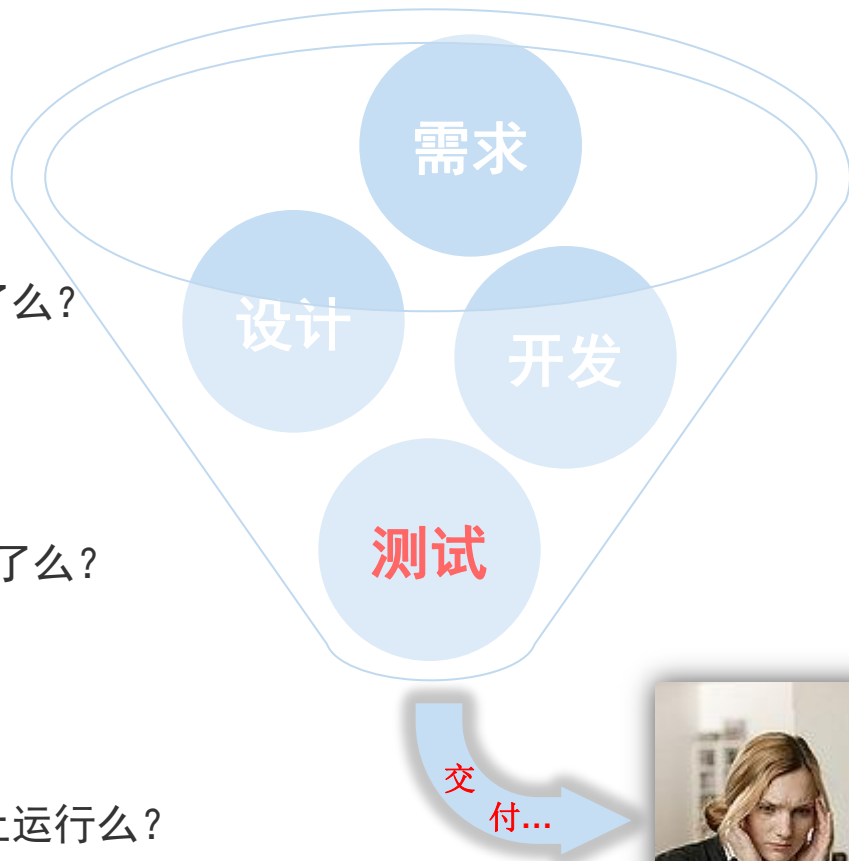


# 敏捷的瓶颈

对于敏捷而言……



- 采用Scrum每日站会就够了吗？
- 能快速响应用户需求就够了么？
- 界面设计可以及时与用户确认就够了么？
- 系统有可扩展的架构就够了么？
- 有合适的快速开发工具就够了么？
- 充分复用已有的技术/业务组件就够了么？
- 每天坚持构建版本就够了么？
- 采用自动化回归测试就够了么？
- 自动化测试用例都能在当天的版本上运行么？



**敏捷需要自动化测试，但自动化测试本身够敏捷么？**

# 艰辛的自动化测试之路

业界有很多优秀的测试工具……

分类	商业	开源
功能测试	QTP、Rational Robot	Selenium Webdriver、RobotFramework
性能测试	LoadRunner、Rational Robot	JMeter

它们都有各自的优点，但也普遍存在的一些问题，让我们举步维艰……

## ➤ 学习成本高

工具的**操作使用**、相关的**脚本语言**、测试过程的**调试分析**，是压在广大技术比较薄弱的测试人员身上的三座大山！

## ➤ 测试脚本维护困难

业界的测试工具本质上还都是**针对页面源码**来编写（或生成）脚本的，**与页面源码高度耦合**、**可读性差**。

页面的任何变化都极有可能**导致测试脚本不可用**，即使提供脚本录制工具，我们能做往往也是重新录制。

## ➤ 断言机制繁琐呆板

测试脚本中的“断言”依赖**手工插入**。页面上蕴含大量的信息，潜在的**断言对象如此之多**、**预期结果时常变化**（甚至是动态的）、**UI样式或UI逻辑**（比如，翻页图标灰显）也很可能出现错误。因此，“断言”可谓是测试人员的“噩梦”！

## ➤ 自定义扩展难度大

测试是个系统的工程，自动化测试是**中间**的一个执行环节。与之关联的工作还有**测试场景设计**、**测试结果分析**、**持续集成**、**版本控制**、**测试用例覆盖率统计**、**测试环境搭建**，等等。

自动化测试工具在扩展方面的局限性，破坏了测试管理的整体性和一致性。

优秀UI框架/工具的采用大大降低了开发成本和难度……

优秀UI框架/工具的采用大大降低了开发成本和难度……

只能输入数字



```

<div id="unisp_form_TextBox_0" class="u-
form-widget widgetid="unisp_form_TextBox_0">
    <div class="u-
form-required dojoattachpoint="requiredNode" style="visibili
ty: visible;">* </div>
    <div class="u-form-field" dojoattachpoint="fieldNode">
        <div class="u-
form-modified" dojoattachpoint="modifiedNode"></div>
        <div class="u-
form-error" dojoattachpoint="errorNode" style="display:
none;"></div>
        <div class="u-form-textbox-field">
            <input class="u-form-textbox-
input" onFocus="unisp_fepk
unisp_fep(this) dojoattachpoint='inputNode,focusMo
de'" style="text-align: right;" maxLength=20"/>
        </div>
    </div>
</div>

```

用例回放的有效性大幅降低，自动化测试变得雪上加霜……

- 页面DOM结构非常复杂——所录制/编写脚本的复杂度变的更大、可读性变得更差；
- 即使页面代码没有任何变化，UI框架的升级也会导致DOM结构的变化——脚本无效的风险变得更大；
- 控件ID是自动生成的，甚至可能随机变化——导致根据ID定位控件的策略无效；
- 为了在不同浏览器下“看着一样”，实际的DOM结构有时也可能不同——测试脚本的兼容性差；

# Web应用的稳定及不稳定因素

用户名  密码  选择去向 部门管理系统

**PSD\_mis**


迭代进展

团队	进展	平均负荷度	偏差
Common Services	39.06%	149.73%	15.96%(延期)
Workflow	19.20%	82.84%	9.21%(延期)
Platform	86.56%	83.06%	8.64%(延期)
Report	84.32%	67.22%	7.26%(延期)
Acilome Integration Test	0%	3.57%	0%

常用链接

- 考勤系统
- RequisitePro
- 资产搜索门户
- 东软网上报销系统
- 云研发环境管理门户
- FTP服务器
- 文档服务器
- 东软ehr系统
- 东软开发者网站
- 知识资产汇聚与交流之地

返回传统主页



用户名:

密 码:

点此试用新首页

## ➤ 稳定因素

用户需求、软件特性

## ➤ 不稳定因素

页面布局、页面样式、用户数据、UI框架及版本

# 像用户一样“测试”软件

“用户使用软件”与“自动化测试软件”之间目前存在一些重要差异……

- 用户操作时只关注页面上能“看”到的，而不用“查看页面源码”；
- 用户会更关注整体业务的正确性、稳定性，而不仅仅是每个孤立页面的功能正确性；
- 用户对页面样式、浏览器兼容性要求越来越高；

如果能像用户使用软件一样进行自动化测试，我们会变得更敏捷……

- 根据界面快速编写测试用例——敏捷应对需求的变化；
- 隔离对技术实现（UI框架、页面样式/布局）的依赖——敏捷应对设计/开发的变化；
- 支持跨浏览器稳定回放——敏捷应对环境的变化；

敏捷的核心是响应变化，  
因此开发和测试都需要快速响应需求的变化；  
而测试额外还需要快速响应开发的变化；

# 聆听自己内心的声音

用户属性		保存		返回	
帐号	autotest2	密码	•••••	姓名	测试2
性别	女	生日	1980-01-01	国籍	中国
证件类型		证件号码		邮件地址	
移动电话		家庭电话		办公电话	
家庭住址					
描述信息					

当你在上述界面上进行操作时，你心里是否会默念：

“账号”输入\*\*\*、“密码”输入\*\*\*、“姓名”输入\*\*\*、“性别”选择\*\*\*、“生日”输入\*\*\*、“国籍”选择\*\*\*，点击“保存”按钮。

类似的，当我们日常使用各种系统时，心里还会默念：

“展开/收拢”树（Tree）的某个节点、关闭某个Tab页、

数据表格（Grid）的下一页/上一页、选中数据表格（Grid）的某一行……

如果测试脚本就像这个样子，大家觉得怎样？

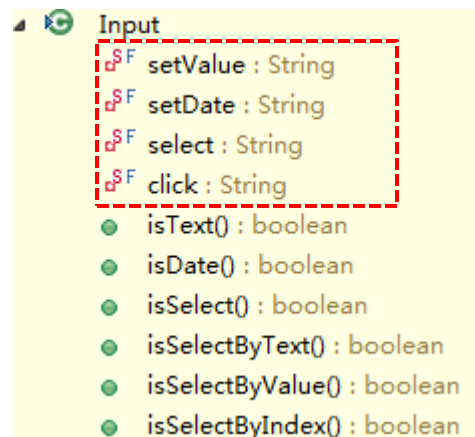
# 用户化的测试脚本

基于界面上可以“看”到的内容定位对象，对象的操作按照用户习惯命名。

## 对象类型

A screenshot of a web form titled "Simple Form". It contains several input fields: "First Name:" with value "yin", "Last Name:" with value "kun", "Company:" with value "neusoft", "Email:\*" with value "test@neusoft.com", "DOB:" with value "03/01/2013" and a calendar icon, "Age:" with an empty field and up/down arrows, and "Time:" with value "8:15 AM" and a dropdown arrow. At the bottom are "Save" and "Cancel" buttons.

## 操作定义



## 脚本示意

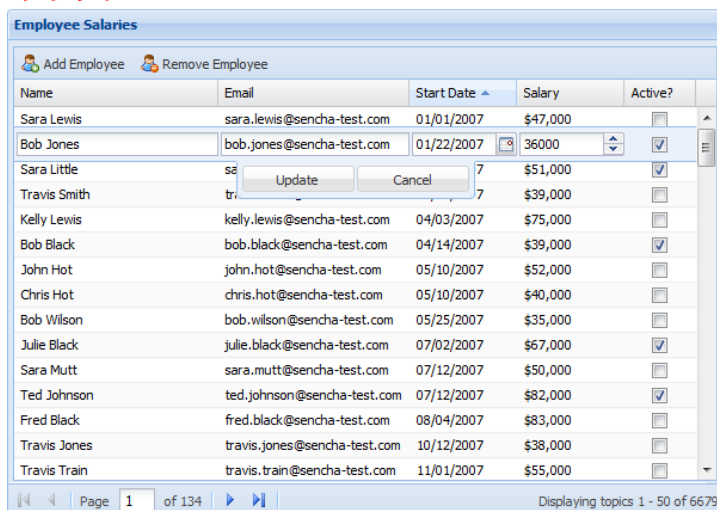
```
<testcase name="输入域示例">
  <method id="表单填写">
    <event id="[input]First Name" name="setValue" value="yin"/>
    <event id="[input]Last Name" name="setValue" value="kun"/>
    <event id="[input]Company" name="setValue" value="东软"/>
    <event id="[input]Email" name="setValue" value="test@neusoft.com"/>
    <event id="[input]DOB" name="setDate" value="2013-03-01"/>
    <event id="[input]Time" name="select" value="8:15 AM"/>
    <event id="[button]Save" name="click"/>
  </method>
</testcase>
```



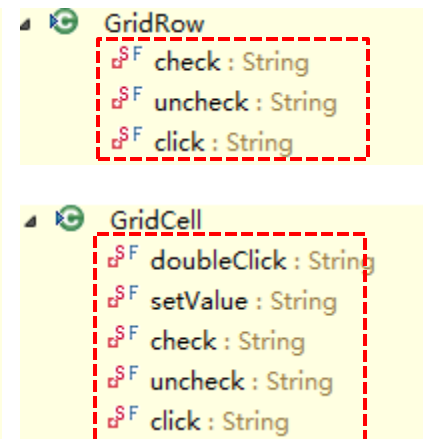
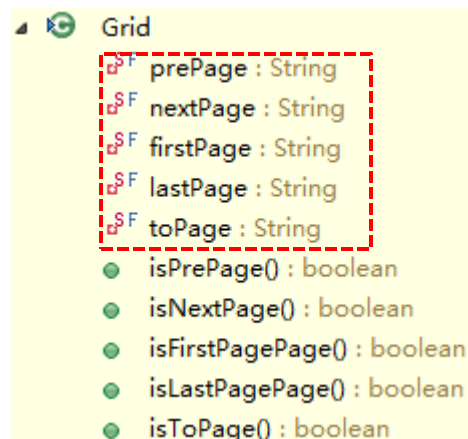
# 用户化的测试脚本

基于界面上可以“看”到的内容定位对象，对象的操作按照用户习惯命名。

## 对象类型



## 操作定义



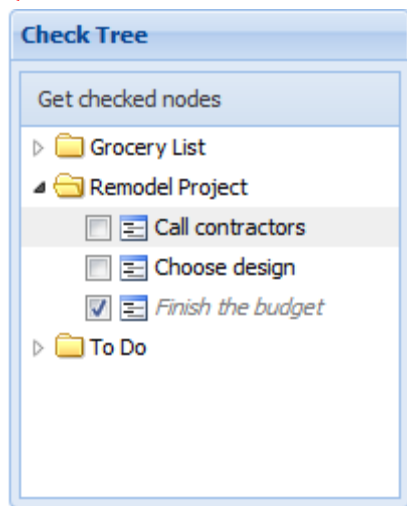
## 脚本示意

```
<testcase name="表格示例">
  <method id="表格操作">
    <event id="[grid]editor-grid" name="nextPage"/>
    <event id="[gridCell]Bill Lee" name="doubleClick"/>
    <event id="[gridCell]Bill Lee,Salary" name="setValue" value="123"/>
  </method>
</testcase>
```

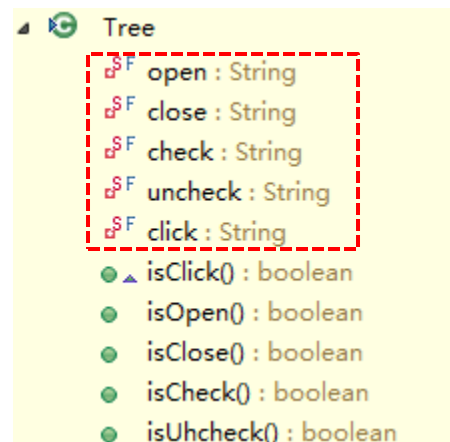
# 用户化的测试脚本

基于界面上可以“看”到的内容定位对象，对象的操作按照用户习惯命名。

对象类型



操作定义



脚本示意

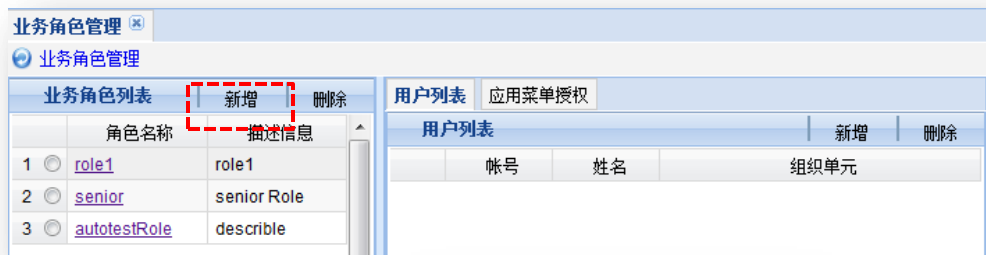
```
<testcase name="树示例">
  <method id="树节点操作">
    <event id="[treeNode]Remodel Project" name="open"/>
    <event id="[treeNode]Finish the budget" name="check"/>
    <event id="[treeNode]Choose design" name="uncheck"/>
    <event id="[treeNode]Call contractors" name="click"/>
    <event id="[treeNode]To Do" name="close"/>
  </method>
</testcase>
```

# 用户化的测试脚本—实例

结合“角色创建及授权”功能，看一下其对应的用户化测试脚本实例：

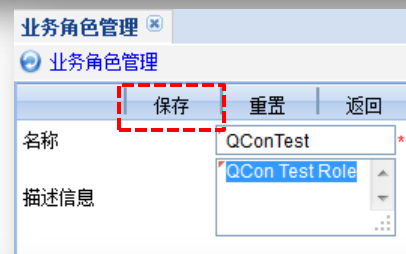
- 点击“业务角色列表”上的“新增”按钮

```
<event id="[titleButton]业务角色列表,新增" />
```



- 输入“名称”和“描述信息”后点击“保存”

```
<event id="[input]名称" name="setValue" value="QConTest" />  
<event id="[input]描述信息" name="setValue" value="QCon Test Role"/>  
<event id="[button]保存" />
```



- 选中“业务角色列表”中“QConTest”对应的行

```
<event id="[gridRow]业务角色列表,QConTest" name="check"/>
```

- 右侧切换到“应用菜单授权” Tab页

```
<event id="[tab]应用菜单授权"/>
```

- 展开“管理控制台”和“组织机构”树节点

```
<event id="[treeNode]管理控制台" name="open"/>
```

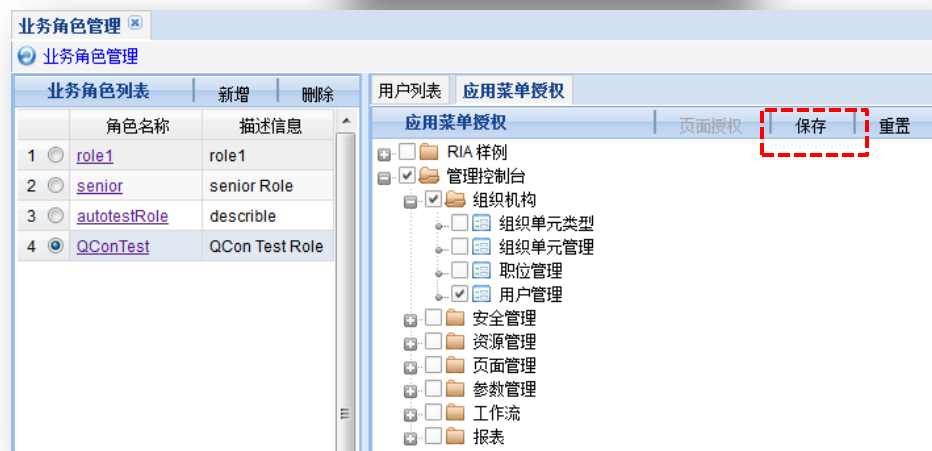
```
<event id="[treeNode]组织机构" name="open"/>
```

- 选中“用户管理”树节点

```
<event id="[treeNode]用户管理" name="check"/>
```

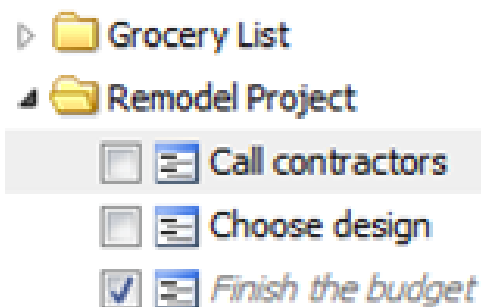
- 点击“应用菜单授权”上的“保存”按钮

```
<event id="[titleButton]应用菜单授权,保存"/>
```



# 用户化的测试脚本—实现

以树节点展开为例，分析用户化测试脚本的实现原理……



“树节点展开”就是对指定节点前面的第一个“展开”图标“点击”操作。

通过复杂XPath可以定位指定节点前面的“展开”图标：

```
//td[contains(@class,'treecolumn')]/*[text()='Remodel Project']/img[contains(@class,'tree-expander')]
```

```
<event id="[treeNode]Remodel Project" name="open"/>
```

用户化的测试脚本中包含了XPath中所有可能变化的信息，

其它信息都可以由UI框架封装的DOM结构决定。

接下来，就是解析XML，然后翻译成XPath了……

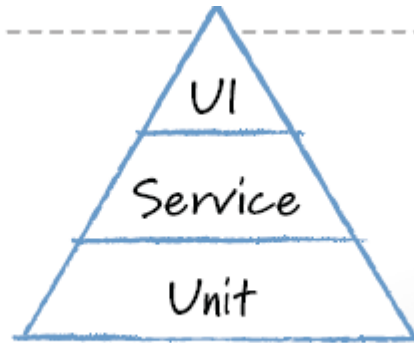
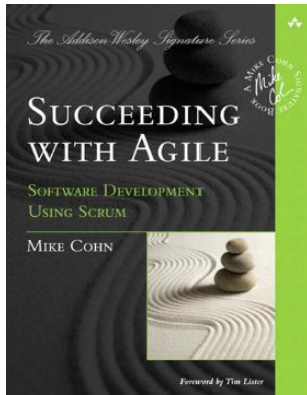


败也“萧何”，成也“萧何”

RIA框架的采用生成了海量的前端源码，导致Web UI自动化测试难度加剧；

但也正是由于RIA框架的采用，让前端代码结构变得一致、规范，为测试脚本封装提供可能性。

# 敏捷中的UI自动化测试实施要点



Automated user interface testing is placed at the top of the test automation pyramid because **we want to do as little of it as possible**. We want this because **user interface tests often have the following negative Attributes:**

- **Brittle**. A small change in the user interface can break many tests...
- **Expensive to write**. A quick capture-and-playback approach to recording user interface tests can work, but tests recorded this way are **usually the most brittle**. Writing a good user interface test that **will remain useful and valid takes time**.
- **Time consuming**. Tests run through the user interface often take a long time to run...

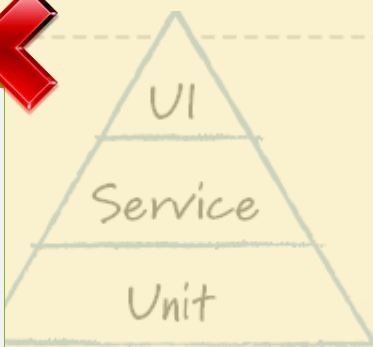
But don't we need to do some user interface testing? **Absolutely ...**

**we no longer need to run all tests through the user interface**. Instead, we run the majority of tests through the service layer... To do this **we need a much smaller set of tests to run through the user interface layer**.

# 敏捷中的UI自动化测试实施要点

## 逻辑错误的想法

UI自动化测试成本高、稳定性差，  
并且专家推荐尽量少做，那我们干脆就不做了吧。  
敏捷主要还是依赖单元测试，那就是开发人员的职责了。



## 积极改善的想法

如何改善UI自动化测试过程中的不利因素，提高稳定性、降低维护成本？  
从而使各层面的自动化测试能够密切配合、相辅相成。



Automated user interface testing is placed at the top of the test automation pyramid because we want to do as little of it as possible. We want this because user interface tests often have the following negative Attributes:

## 我们的实践

Neusoft®

- 基于业务测试框架开展自动化测试，提高UI自动化用例的稳定性，并降低其维护成本；
- UI自动化用例覆盖基本业务流程（不做异常条件和边界值测试），对前端校验编写独立测试用例；
- 测试团队同时也参与服务层的自动化测试，并与UI层自动化测试统一管理、相互补充；
- 测试团队从UI层入手开展自动化测试，培养每日构建文化，进而带动开发人员单元测试的积极性；

We no longer need to run all tests through the user interface. Instead, we run the majority of tests through the service layer... To do this we need a much smaller set of tests to run through the user interface layer.

Neusoft



- 技术发展对自动化测试的挑战
- 提高用户界面测试的敏捷程度
- 测试用例的高复用与自动装配
- 云计算助力测试环境高效管理



# 测试数据参数化—测试用例与数据解耦

Sprint管理 工作回顾

Sprint管理 > Backlog管理

Backlog列表 上移 下移 Backlog信息

提供与测试脚本描述方式匹配的测试报告  
提供简易的测试用例编辑器及运行视图  
Web端功能规划及需求分析

backlog名称 提供简易的测试用例编辑器及运行视图  
编号 S2  
负责人 盛营  
预估工作量 316  
实际工作量 74

TaskForm

Task信息

名称 提供UTF运行、校验视图  
工作量(人时) 80  
难度 B  
活动类型 编码实现

保存

Sprint管理 填写日报

填写日报

待办任务 申请任务... 取消申请 完成任务 新增日报 保存 保存

任务名称	Backlog名称	职能类别	剩余工	计划解决时
1 提供UTF运行、校验视图	提供简易的测试用例...	研发	40	
2 UTF日常支持及研发管理		测试		
3 UniEAP日常测试相关工作		测试		
4 Aclome日常测试相关工作		测试		
5 新员工入职培训		策划		
6 参数化需求分析及设计	支持基于DB的数据驱...	策划	17	
7 代码检查改善专项		管理		

对应任务 提供UTF运行、校验视图  
日期 2013-09-15  
所用时间(人时) 8  
工作内容描述 补充根据语法API生成Schema的功能

Sprint管理 工作回顾

工作回顾

Task信息

工作回顾日期 2013-09-13 查看有工作量的任务

Task名称	Backlog名称	任务状态	参与人	预估工作	实际工作	剩余工作	工作内容描述
1 提供UTF运行、校验视图	提供简易的测试用例...	进行中	殷坤,盛营	80	21	40	提供“UTF>运行”...
2 HTML格式测试报告生成	提供与测试脚本描述方...	进行中	戚永建	24	154	4	根据报告内容、原...
3 测试报告内容及样式原...	提供与测试脚本描述方...	已完成	殷坤	8	6	0	用Excel设计测试...
4 测试用例整体合法性校验	提供简易的测试用例编...	移除	殷坤	40	0	0	支持测试用例运行...
5 界面设计	提供简易的测试用例编...	已完成	盛营,殷坤	16	14	0	脚本编辑工具的界...
6 TestNG报告自定义技术...	提供与测试脚本描述方...	已完成	戚永建	32	56	0	了解TestNG自定...
7 支持在TestNG报告实现...	提供与测试脚本描述方...	已完成	殷坤	16	11	0	让业务脚本语法与...
8 脚本编写语法提示	提供简易的测试用例编...	已完成	盛营	80	39	0	支持根据控件类型...

## 业务场景

测试用例中的输入数据发生变化的可能性很大，并且用例中不同位置输入的数据是相同的。  
因此需要把测试数据以参数的形式提取出去。

## 脚本示例

```
<event id="[input]名称" name="setValue" value="${TaskName}"/>  
<event id="[gridRow]代办任务,${TaskName}" name="check" />  
<event id="[gridRow]Task信息,${TaskName}" name="check" />
```

	A	B
1	参数名	参数值
2	TaskName	提供UTF运行、校验视图
3	TaskType	编码实现
4	TaskDescribe	提供“UTF>运行”、“UTF>校验”的菜单及运行/校验结果视图。
5		
6		

Common 任务管理 创建用户 创建业务

## 测试数据参数化—参数动态赋值

首页

单位参保登记

单位基本信息

单位参保及银行账户信息

单位基本信息

单位编号	127649	单位名称	自动化测试使用的单位名称		
单位类型	企业	经济类型	内资	隶属关系	中央
行业代码	农、林、牧、渔业	组织机构代码	94101001-9	单位状态	登记名称
行业风险类别	制造业—轻工业（风险较小行业）	组织机构代码		增值税机关	按照一步骤
邮政编码					
经办机构	养老保险费征缴处				
单位传真				登记有效期限(年)	
执照(证)号码				执照发证机构	
批准成立单位		批准日期		批准文号	
法人姓名		法人身份证号码		法人电话	12345678

单位信息修改	
<b>单位基本信息</b>	
单位编号 127040	单位名称
单位类型 企业	经济类型 内资
行业代码 农、林、牧、渔	组织机构代码 91110101-9
行业风险类别 资产一般（类属中小企业）	主管部门 中国辽宁大连
注册地址 中国辽宁省大连市	单位地址 市本级
行政区域 市本级	执照(证)种类
执照(证)日期	登记有效期(年)
批准成立单位	执照年检标志
法人姓名	
上级单位编号	
社保登记代码	
经费来源	
税务机构编号	
<b>联系信息</b>	
联系人姓名	办公电话

单位基本信息		单位参保及银行账户信息	
单位基本信息			
单位编号	127649	单位名称	自动化测试使用的单位名称
单位类型	企业	经济类型	内资
行业代码	农、林、牧、渔业	隶属关系	中央
行业风险类别	行业一（其他较小行业）	组织机构代码	91101001-9
注册地址		单位状态	正常单位
经办机构	单位自行申报局	主管部门	地税征收机关
单位传真		组织机构代码	地税一分局
执照(证)号码		行政区划代码	北京市
批准成立单位		执照(证)种类	
法人姓名		发照(证)日期	
上级单位编号	127649	批准日期	
社保登记证编号		法人身份号码	
经费来源		特殊企业标志	非普通企业
税务机构编号		备注	
		财政供养单位	
		区属下挂标志	
		登记有效期限(年)	
		执照发证机构	
		批准文号	
		法人电话	12345678
		大额代扣标志	
		一次性缴费单位标志	
联系人信息			
联系人姓名	办公电话	电子邮箱	移动电话
联系人类型	地址	创建日期	传真
所在部门			

## 业务场景

- 新增“单位参保登记”时，“单位编号”是系统自动分配的；
- 修改“单位参保信息”时，需要根据“单位编号”进行查询；

## 脚本示例

```
<event id="[input]单位编号" name="setParam" value="${单位编号}" />
```

# 测试数据参数化—参数支持分组

用户名:  
admin

密 码:  
●

Login

用户管理

用户管理 > 用户属性

用户属性 保存 重置

基本属性

帐号: autotestUser 密码: 生日: 1983-09-14 姓名: 中

性别: 男 证件类型: 证件类型: 证件号码: 邮件地址: 办公电话: 家庭住址: 办公电话: 传真: 描述信息:

## 业务场景

- 先用管理员登录，创建一个用户，并为其授权；
- 再换由新增的用户登录，进行后续业务操作；

## 脚本示例

### 测试场景定义

```
<test-suite path="console/登录/登录.xml" paramGroup="admin"/>  
<suite-file path="console/组织机构/用户管理/创建用户.xml" paramGroup="user"/>  
<suite-file path="console/安全管理/业务角色授权.xml" paramGroup="user"/>  
<suite-file path="console/登录/注销.xml" />  
<suite-file path="console/登录/登录.xml" paramGroup="user"/>
```

### 测试用例定义

```
<method id="登录">  
  <event id="用户名" name="setValue" value="${userAccount}"/>  
  <event id="密码" name="setValue" value="${userPassword}"/>  
  <event id="Login"/>  
</method>
```

## 业务角色管理

### 业务角色管理

	角色名称	描述信息
1	role1	role1
2	senior	senior Role
3	autotestRole	describe
4	QConTest	QCon Test ...

### 用户列表

#### 用户列表

	帐号	姓名	组织单
1	autotestUser	autotestUser	

用户名:  
autotestUser

密 码:  
●

Login

	A	B	参数组
1	参数名	参数值	参数组
2	userAccount	admin	admin
3	userPassword		1 admin
4	userAccount	autotestUser	user
5	userPassword		1 user
6	业务角色名称	autotestRole	user

# 测试数据参数化—根据参数值微调脚本行为

用户管理

用户管理 > 用户属性

用户属性 | 保存 | 重置 | 返回

基本属性

帐号: autotest \* 密码: \* 姓名: autotestName \*

性别: 男 生日: 1982-05-07 国籍: 中国

证件类型: 身份证 证件号码: 34212145983014514 邮件地址:

移动电话: 家庭电话: 办公电话:

家庭住址: 传真:

描述信息: 自动化测试增加的用户

用户管理

用户管理 > 用户属性

用户属性 | 保存 | 重置 | 返回

基本属性

帐号: autotest \* 密码: \* 姓名: autotestName \*

性别: 男 生日: 1982-05-07 国籍: 中国

证件类型: 军官证 证件号码: SS0123453554558 邮件地址:

移动电话: 家庭电话: 办公电话:

家庭住址: 传真:

描述信息: 自动化测试增加的用户

## 业务场景

“新增用户”和“修改用户”是复用同一个页面，在实际操作时，新增用户会录入比较完整的信息，而修改用户时往往只涉及其中个别字段。

## 脚本示例

### 测试场景定义

```
<suite file="console/组织机构/用户管理.xml" paramGroup="addUser"/>  
<suite file="console/组织机构/用户管理.xml" paramGroup="modifyUser"/>
```

### 测试用例定义

```
<test id="用户信息维护">  
  <event id="[input]帐号" name="setVal" value="${userAccount}"/>  
  <event id="[input]密码" name="setVal" value="${userPassword}"/>  
  <event id="[input]姓名" name="setVal" value="${userName}"/>  
  <event id="[select]性别" name="setText" value="${性别}"/>  
  <event id="[date]生日" name="setVal" value="${生日}"/>  
  <event id="[select]国籍" name="setText" value="${国籍}"/>  
  <event id="[select]证件类型" name="setText" value="${证件类型}"/>  
  <event id="[input]证件号码" name="setVal" value="${证件号码}"/>  
  <event id="[input]描述信息" name="setVal" value="${描述信息}"/>  
  <event id="[button]保存"/>  
</method>
```

	A	B	C
1	参数名	参数值	参数组
2	userAccount	autotest	addUser
3	userPassword		1 addUser
4	userName	autotestName	addUser
5	性别	男	addUser
6	国籍	中国	addUser
7	生日	1982-05-07	addUser
8	证件类型	身份证	addUser
9	证件号码	342121459830145147	addUser
10	描述信息	自动化测试增加的用户	addUser
11	证件类型	军官证	modifyUser
12	证件号码	SS0123453554558	modifyUser

# 测试场景定义—支持指定要执行的测试方法



## 业务场景

一个页面上通常会包含多种操作，不同操作之间往往存在大量重复的前置动作（比如，点击菜单、选择记录）或后置动作（比如，修改输入项）。

## 脚本示例

### 测试场景定义

```
<suite-file path="console/组织机构/用户管理.xml" include="增加,用户信息维护" paramGroup="addUser"/>
<suite-file path="console/组织机构/用户管理.xml" include="选择用户,修改,用户信息维护" paramGroup="modifyUser"/>
<suite-file path="console/组织机构/用户管理.xml" include="选择用户,删除" paramGroup="delUser"/>
```

### 测试用例定义

```
<method id="页面初始化" alwaysRun="true">
  <event id="[menu]组织机构.用户管理"/>
</method>
<method id="选择用户">
  <event id="[gridRow]用户列表,${userAccount}" name="check"/>
</method>
<method id="增加">
  <event id="[button]新增"/>
</method>
<method id="修改">
  <event id="[gridCell]用户列表,${userAccount},帐号" name="click"/>
</method>
<method id="删除">
  <event id="[button]删除"/>
  <event id="[button]确定"/>
</method>

<method id="用户信息维护">
  <event id="[input]帐号" name="setValue" value="${userAccount}"/>
  <event id="[input]密码" name="setValue" value="${userPassword}"/>
  <event id="[input]姓名" name="setValue" value="${userName}"/>
  <event id="[select]性别" name="setText" value="${性别}"/>
  <event id="[date]生日" name="setValue" value="${生日}"/>
  <event id="[select]国籍" name="setText" value="${国籍}"/>
  <event id="[select]证件类型" name="setText" value="${证件类型}"/>
  <event id="[input]证件号码" name="setValue" value="${证件号码}"/>
  <event id="[input]描述信息" name="setValue" value="${描述信息}"/>
  <event id="[button]保存"/>
</method>
```

## UI自动化测试用例的设计粒度

Web应用中最直观的组织单元是“菜单”，所以测试用例也可以以“菜单”为粒度进行设计。即，每个菜单对应一个测试用例文件，每个测试用例文件包含多个测试方法（method），通过“测试数据参数化”和“测试场景中支持配置测试方法”使测试用例被更大程度的复用。

# 测试用例组织—支持根据组件依赖关系自动装配

配置 > 应用管理

名称 ^			版本号	操作
aclome-app-common-monitor			4.5.0	■ ×
aclome-app-db			4.5.0	■ ×
aclome-app-db-db2			4.5.0	■ ×
aclome-app-db-mysql			4.5.0	■ ×
名字	作者	描述		
aclome-mysql-keyreadratio-mo...	liuxinning	读请求磁盘命中率		
aclome-mysql-temporarydisktabl...	hanbq	磁盘临时表数量		
aclome-mysql-bytessent-monitor	hanbq	网络流出流量		
aclome_app_db_mysql_tablescan	chuk	tablescan		
aclome_db_mysql_connectionco...	liuyoutao	连接数		
aclome-mysql-temporarytables-...	hanbq	内存临时表数量		
aclome-mysql-keywriteratio-mon...	liuxinning	写请求磁盘命中率		
aclome_app_db_mysql_cache	chuk	缓存信息		
aclome-mysql-bytesreceived-mo...	hanbq	网络流入流量		
aclome-app-db-oracle			4.5.0	■ ×
aclome-app-db-sqlserver			4.5.0	■ ×
aclome-app-distribute-datacenter			4.5.0	■ ×
aclome-app-general-service			4.5.0	■ ×
aclome-app-general-service-ftp			4.5.0	■ ×

本地安装 资产库装配

## 业务场景

在组件化开发模式下（比如，利用OSGi技术），当组件被停止/移除之后，该组件对应的功能不是简单的从页面上隐藏，而是从ClassLoader或部署目录中被彻底删除。

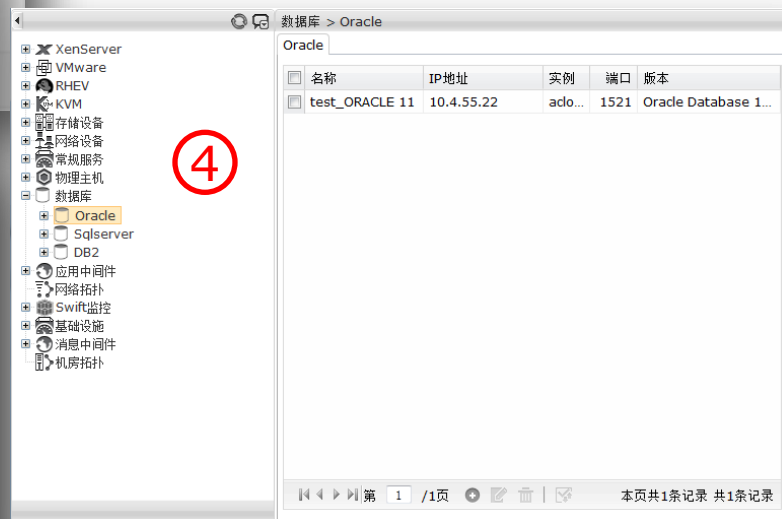
由于大部分组件都包含服务端、客户端、数据库端资源，所以在开发过程中很难保证所有资源都严格遵守设计时的依赖关系，因此部分组件移除后，系统其它组件的功能会存在一定的风险。

# 测试用例组织—支持根据组件依赖关系自动装配



## 场景示例

- ① 没有组件被移除时，全局资源搜索功能会从所有类型数据库中检索结果、数据库类型包含Mysql；
- ② 移除Mysql组件；
- ③ 全局资源搜索、数据库类型列表中均不包含Mysql类型；
- ④ Oracle等其它类型资源的各种功能应该不受任何影响；





# 测试用例组织—支持根据组件依赖关系自动装配

## 脚本示例

➤以参数化的形式定义几组需要被移除的组件，在用例执行之前用**管理员登录停止这些组件**；

```
<suite-file path="系统/login.xml" paramGroup="sysAdmin"/>
<suite-file path="配置/应用管理/组件装配.xml" paramGroup="stopComps1"/>
```

	A	B	C
1	参数名	参数值	参数组
2	components	aclome-app-db-mysql;	stopComps1
3	components	aclome-app-db-mysql;aclome-app-db-db2	stopComps2
4	components	aclome-app-db	stopComps3
5	components	aclome-app-alert	stopComps4
6			
7			

➤在测试场景定义文件可以为测试用例指定所对应的组件，当对应的**组件未被启动时**，相应的用例不执行；

```
<suite-file path="资源库/数据库/Mysql/监控DB维护.xml" component="aclome-app-db-mysql"/>
<suite-file path="配置/指标和策略配置/指标配置.xml" component="aclome-app-db-mysql" paramGroup="mysql"/>
<suite-file path="告警/告警管理.xml" component="aclome-app-alert"/>
<suite-file path="资源库/数据库/Oracle/监控DB维护.xml" component="aclome-app-db-oracle"/>
<suite-file path="配置/指标和策略配置/指标配置.xml" component="aclome-app-db-oracle" paramGroup="oracle"/>
```

➤在测试用例定义文件可以为测试方法指定与测试用例不同的组件，当对应的组件未被启动时，**相应的方法不执行**；

```
<method id="告警查询" component="aclome-app-alert">
  <event id="[tab]告警"/>
  <event id="[radio]周" name="setValue" value="${dbName}"/>
</method>
```



## 用例装配范围

理论上可以自动获取所有组件的依赖关系，然后穷举启停组合，进行自动装配测试，但要求测试用例的模块化程度非常高，并且执行耗时太长。

目前建议预设一些常用的装配组合进行日常自动化测试，并且根据客户需求随时补充装配组合。



- 技术发展对自动化测试的挑战
- 提高用户界面测试的敏捷程度
- 测试用例的高复用与自动装配
- 云计算助力测试环境高效管理

# 云测试之我见



Web自动化测试三要素

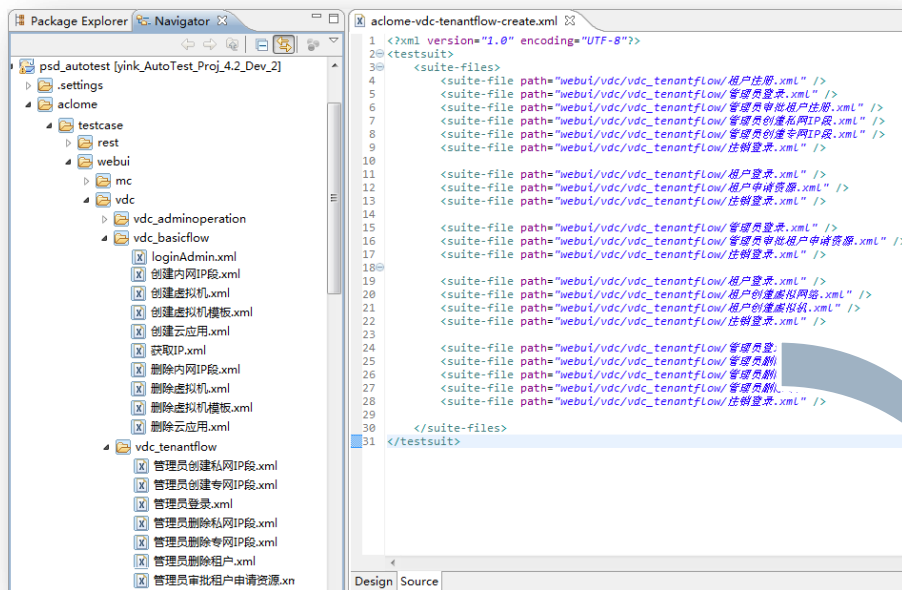


## 云测试

是云计算技术在自动化测试领域的应用，主要目的是为了提升测试脚本、测试环境（客户端、服务器等）的管理效率，降低总拥有成本。

# 云测试—测试脚本管理

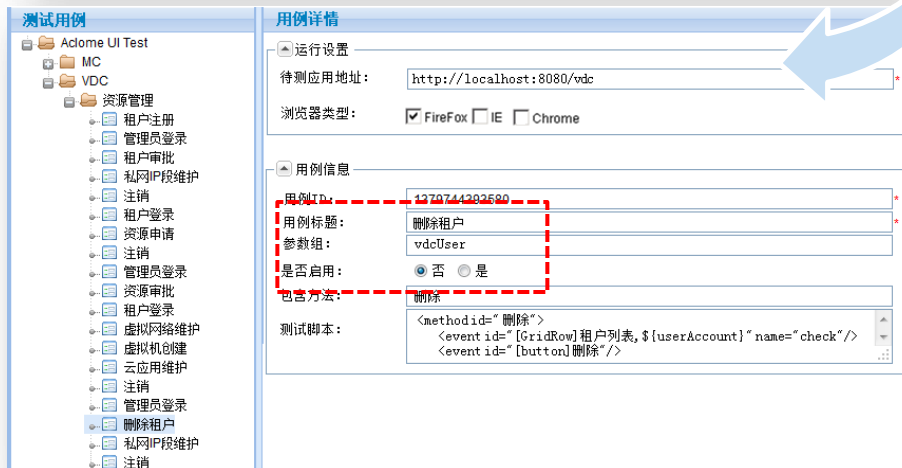
本地测试工程中的脚本



把稳定、常用的测试脚本部署到远程云测试平台上，用户无需下载任何测试脚本、无需安装任何测试工具，即可将选择的脚本在指定的待测应用上执行。

通过把测试脚本变成“服务”，便于除测试之外的其他团队复用，充分发挥自动化测试的附加价值。

可在线执行的测试脚本

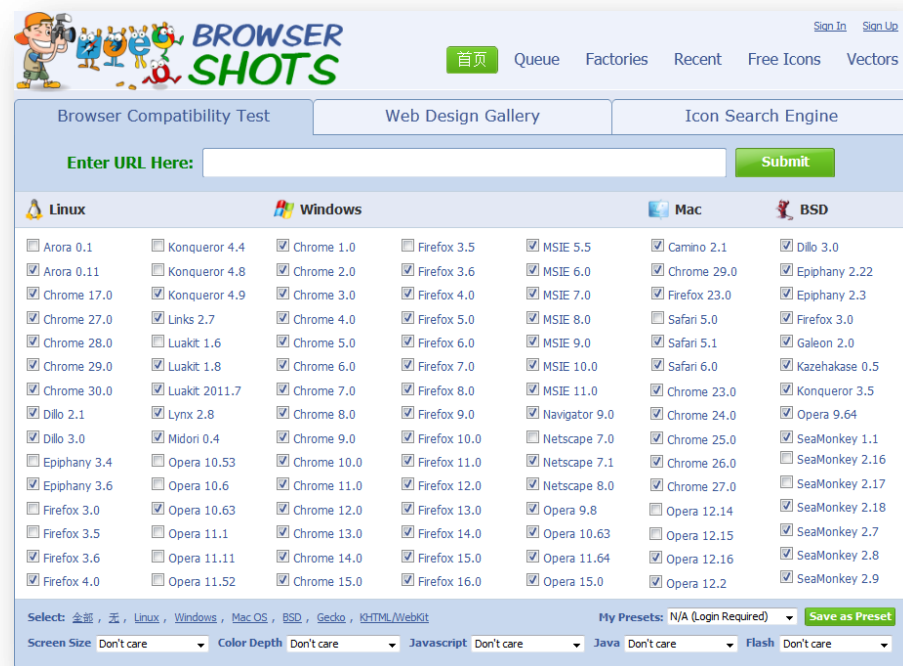


# 云测试—客户端管理

由于本地计算机能安装的浏览器类型有限，所以在对Web应用进行浏览器兼容性测试时，往往需要创建多个虚拟机用以安装各种浏览器。

出于对网络连通性和数据安全性等因素的考虑，对于有大量浏览器兼容性测试需求的企业，建议搭建企业私有云测试平台来提升这方面的管理效率。

国内一些主要提供移动终端设备的云测试产品也属于此类应用。



Neusoft

# 云测试—服务器管理

虚拟机申请表

数据编号:

序号	申请人	申请日期	开始日期	结束日期	组织	产品方向	用途描述	设备名称	设备描述	规格配置	CPU (Core个数)	内存 (GB)	硬盘 (GB)	网络	IP
1	张静林	2011/12/2	2011/12/2	2012/1/1	测试	所有	测试	高配	高配	高配	2	20	20	10.4.44.44	
2	张静林	2011/12/2	2011/12/2	2012/1/1	测试	所有	测试	高配	高配	高配	2	20	20	10.4.44.44	
1	张静林	2011/12/2	2011/12/2	2012/1/1	测试	所有	测试	高配	高配	高配	2	20	20	10.4.44.47	
1	张静林	2011/12/2	2011/12/2	2012/1/1	测试	所有	测试	高配	高配	高配	2	20	20	10.4.44.49	
1	张静林	2011/12/2	2011/12/2	2012/1/1	测试	所有	测试	高配	高配	高配	2	20	20	10.4.44.51	
1	张静林	2011/12/2	2011/12/2	2012/1/1	测试	所有	测试	高配	高配	高配	2	20	20	10.4.44.56	
1	张静林	2011/12/2	2011/12/2	2012/1/1	测试	所有	测试	高配	高配	高配	2	20	20	10.4.44.61	
1	张静林	2011/12/2	2011/12/2	2012/1/1	测试	所有	测试	高配	高配	高配	2	20	20	10.4.44.73	
2	张静林	2011/12/2	2011/12/2	2012/1/1	测试	所有	测试	高配	高配	高配	2	20	20	10.4.44.74	

封面：更改履历 | 虚拟机申请表 | 虚拟机变更记录表

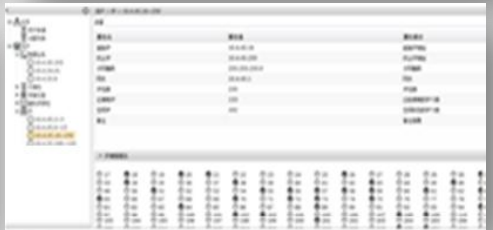
管理效率低

设备利用率低

## 云测试环境管理平台关键特性

- 资源申请、审批、分配、变更、回收；
  - 支持虚拟机创建及模板转换；
  - 支持多租户管理；
  - 支持对资源利用率的实时监控与多维分析；
  - 支持对计算、存储及网络资源统一管理；
- 【注】虚拟机若用于性能测试，务必确认能够已独占方式分配计算资源。

云测试环境管理平台



Neusoft

# 回顾

自动化测试成败的关键在测试脚本的维护成本和运行效率

更适合敏捷

用户化的脚本语法，符合行为习惯、  
隔离对页面源码的依赖。

更便于复用

测试数据参数化，并支持分组和动态赋值；  
测试用例组件化，并支持灵活、自动装配；

更简化运行

降低测试脚本、测试环境的获取及维护成本。



# Neusoft

Beyond Technology

Copyright © 2011 by Neusoft Corporation. All rights reserved.

**Neusoft**