目录

一个软件测试工程师的学习体验.....2

基于Agere开发平台的GSM手机自动化测试解决方案.....7

ISO和CMM，我们该选择谁呢？.....19

性能测试工具之研究.....25

关于SQAGetProperty的使用.....40

针对办公自动化系统软件的测试分析方法.....49

性能测试原理及性能测试实例分析.....54

软件测试和第三方测试服务的需求调查.....66

浅谈软件测试自动化解决方案.....80

智能测试自动化.....87

软件测试园地.....101

一个软件测试工程师的学习体验

夏梦远

【摘要】软件质量越来越受到人们的关注，软件测试作为新兴行业有很多不完善的地方。很多从事软件测试工作的同行处于迷茫之中，如何提高，如何解决测试工作中的实际问题，困惑着每一个人。本文总结了一下个人经验，希望对大家有帮助。

【关键词】软件测试 软件测试学习 软件测试工程师

我最初参加测试工作的时候，不知道什么是软件测试，集成测试和系统测试的概念经常混淆，CMM 是什么就更加不知道了。那时候最简单的开关机也是通过直接拔插电源完成，安装系统对我来说简直是有史以来人类的最高技能，对于那些拿着螺丝刀安装机器的人就认为是宇内超级高手，身具杀人于无形之绝世秘技。拿破仑说不想当将军的士兵不是好士兵，我最初的梦想就是想成为软件测试的高手，傲视天下。所以不断偷师，总结经验，自认为掌握了成为高手的几个秘技，这几年混迹“江湖“还算无往而不利。不敢独享，望与吾辈测试人员切磋，早日总结成功密技之大成，助新进人员早日入门，也算不愧对东北活雷锋的称号。

第一招 学会利用网络

刚参加工作面对浩瀚的网络世界，当时如刘姥姥进大观园，什么都新奇，什么都想要，从网上下载很多源程序的代码，软件技术文档之类，恨不得把所有的好东西收集到手中，其实有些在他人看起来就是垃圾一堆。当时觉得有了这些“武林秘籍“，成为高手指日可待。最初参加工作由于自己工作努力有幸转为开发，加入项目组后我的习惯还是没有改，反而变本加厉，手中的资源更加多，上网的时间更加频繁。

一次项目经理分配任务，觉得依靠手中的秘籍加上自己的“聪明才智“很快会完成，不料短短的时间，所有的一切变成了马奇诺防线。解决问题很慢，思路不清晰，项目经理在对我施压的过程中教会了我终身难忘的一招，学会利用网络寻找要解决问题的答案，从此 Google 成了我的最爱，关键字成了我变化的招数。在软件测试工作中，他帮我解决了很多疑难问题，解答了很多令我迷惑的地方。也是我帮助测

试同行解决问题手段之一，很多软件测试新手，甚至老手都没有意识到自己手上就握有“无敌秘籍“，所以只要你耐心找，答案就在身边。

这里总结一下利用网络搜索引擎的技巧：

1. 组合搜索

每次搜索某个文件，如果只给出一个单词进行搜索，经常会出现成千上百万计的匹配网页。然而如果再加上一个单词，那么搜索结果会更加切题。

2. 选择表述内容的词组

一般我在网页搜索引擎的时候，选择一些可以表达我要查找内容的关键词组，用来缩小搜索范围，从而找到搜索结果是最好的办法。运用词组搜索涉可以先先简单地输入一个问题作为词组搜索，如果仍然找不到合适的，那就用多个可以表达要查询内容的关键字进行查询。

3. 定位信息来源

有的时候用词组搜索不到或者无法准确表达所需信息。可以用另一种方法直接到信息源，就是直接到到提供某种信息的站点去。可以用公式“www.公司名.com”去猜测某一组织的特点。从而得到所要搜索的信息的主要词组

其实网络上还有很多关于搜索技巧的文章，大家可以自行学习。千万要记住搜索引擎是帮助你成功的有力武器。

第二招　学会动手

参加软件测试工作后，随着工作经验的增长自我感觉越来越好。在公司里也逐渐受到同事领导的重视，一次针对公司的新的软件功能进行测试的时候，像往常一样“随手“测试出了几个 Bug，然后“仔细“的填写了 Bug 单（这个 Bug 的现象已经出现了很多次了）。这时候测试经理走过来，重新复查了一下填写的 Bug。他在重现我的 bug 的过程中，简化了我的输入变化，bug 神奇的又出现了，同样的现象，他关闭软件重新变化输入，扩展出 10 几个变化后，软件不动了，内存不断上升。终于他找到了产生软件的 Bug 的原因，然后对我说“寻找 Bug 要准确定位，我们开发团队是一个整体，时间是等量的，时间不在你身上浪费，就是在他身上浪费。如果测试人员每次发现的 bug 描述不

清楚，并且多个问题潜在的错误原因是一个，虽然操作可能稍微有些变化。这样开发人员在重现 bug 的时候他要调试跟踪判断，很花费时间，而且效率低。如果测试人员发现 bug 的时候多动手可以更加准确的定位 bug 步骤和原因，给开发人员最精确的步骤和准确的描述，这样整个团队才能高效，所以需要大家协作!。“ 。

在以后的日子里，每次解决问题的时候我都记得多试验几次，多尝试。网上很多朋友还有同事问我问题的时候，其实他们只是万里长征就差一步，只要再多动手实验一次就可以达到目的了。所以多动手，多尝试。

第三招　思考自己所作的

刚开始入行的时候，总是思考如何做好软件测试。认为公司的测试流程混乱总是很郁闷，认为自己学不到东西，如何才能测试好产品，常说心动不如行动，以前看到古龙小说中经常出现的场景无名小子不断挑战高手，总结积累。我总结了有些经验是实战中得到的，所以不断尝试引入新的测试流程然后评估，这个过程虽然很痛苦，但是从中积累了不少经验。这段时间让我学习到了很多东西，接触了 ISO,CMM，测试管理工具，自动化工具（因为公司不正规给了我很多学习的机会，后来到了比较大的软件公司后，以前的经历给了我更多的发展机会，因为大公司非常正规了，公司内部人员分工明确，所以能力的锻炼反倒少了）。由于工作中经常写报告反倒养成了总结教训的习惯，因为纸面上的东西是永远也忘不掉的。在写的过程中可以不断补充扩展，整个过程是思想升华的过程，当年达摩面壁九年就是融会贯通的典型例子，如果他不是有个思考的过程，他也不能成为一代大家。如果后来不时有人把他的绝技记录下来，也就不能有后来的少林寺七十二绝技。

所以善于思考，总结经验，也是成为高手之路的不二法决。

第四招　学会利用论坛资源

其实测试新兵和测试高手之间的区别，往往是不会利用现有资源。在论坛中我们会看到很多新手不断的提问，但是有很多问题其实都是已经别人提过了，或者已经有解决方案的。所以经常会看到“测试高手“的身影，并且不提问题，而且还能“锄强扶弱“，是测试新丁的救命稻草。好像是高手们无所不能，其实摘掉这层耀眼的光环，他们并没想像得那么厉害，只不过通过自己的搜索找到的答案，然后

帮助其他人。当然也有很多人都是通过自学，然后在论坛中交流得到了很多经验，高手其实也是因为善于思考问题，亲自动手解决问题。所以动手和利用论坛资源的过程中他们也在不断提高。

很多时候看到论坛中有人提问，问题描述不清，很多人看了很困惑。发贴题目动不动请高手帮忙，救命之类的，好像天下大乱，世界末日。虽然这个题目很招人，但是无法让那些想帮助你的人帮你，因为题目不清晰，而且高手字样吓阻了很多人。其实问问题也是个思路整理的过程，描述清晰，让人理解清楚，才能望文知意知道你的当前发生问题的环境，才能让那些想帮你的人解决问题，否则给人无从下手的感觉，解决问题效率不高。

第五招　学习和你所测试的软件产品相关的知识

要想成为好的测试人员，还要了解你要测试的软件的相关知识。要了解软件产品的架构是什么样的。要了解软件的市场需求，在接触软件之初要可以多看看用户的反馈信息，这些才是用户最关心的，也是你在测试中需要注意的问题，满足客户是最大的需要。但是了解软件需求之后要学会要多读些软件系统的技术文档，软件设计文档，这些文档可以帮助你了解产品如何工作。还有多看看公司 Bug 库中的问题，这些存在的问题可以帮助你了解软件产品那些地方存在缺陷，软件系统那些地方会出现错误。软件是运行在一个大环境中，如果对系统不熟悉，那么有些问题你不能从一个更广阔的层面考虑，学习操作系统的知识，有助于你发现缺陷，定位问题更加准确。比如软件运行在 Windows 或者 Linux，如果你不懂操作系统，你就无法建立测试环境，有些时候时候软件的组件发生问题，就是你系统配置造成的，对系统不熟悉，你会把外在原因归结为软件本身。所以要学习关于和软件系统相关的知识，比如编程，网络，数据库等。不一定你要学习到多好的程度，只是通过这些扩展的知识面，你可以在发现问题，解决问题上不会局限在狭小的圈子里。

和一切相关的人员交流，不同的交流渠道，获取消息是不同的，角度也不同。和客户交流，你会在测试中从客户的角度发现问题；和开发人员交流，你会了解开发人员怎么实现软件功能的；和项目管理人员交流，你会知道开发进度以及遇到的困难。

以上五招伴我走过多年，让我领略到测试的乐趣。成为高手并不难，其实两招就可以学会利用资源，学会动脑。在平常工作中时时刻刻将学习的精髓融入其中，将“精髓”转化为“物质”，转化为扎扎实实

的、实实在在的行动。独孤九剑变化无穷，第一等的剑法，胜在手中无剑，心中有剑。成为高手不是没有可能，世界上没有不可能的任务。

基于 Agere 开发平台的 GSM 手机自动化测试解决方案

毛宏才

【摘要】本文就杰尔系统（Agere system）平台基础上开发的 GSM 手机自动化测试提供一些技术介绍，并结合实际例子讲解一些应用经验，来说明自动化测试在手机功能测试一级中所带来的效率。

【关键词】手机平台，杰尔系统，Trace，PTE 命令，手机软件功能测试，自动化测试

一、国内手机功能测试现状：

当前国内手机厂商和设计公司据统计已达到 300 多家，但至今所有的设计开发都是基于国外技术平台基础上的二次开发，即通常所说的 MMI 开发， 提供开发的手机平台目前主要有德州仪器（TI），英特尔（Intel），飞思卡特（freescale），杰尔系统（Agere system），英飞凌（infineon），瑞萨科技（renesas），菲利普半导体（philips），意法半导体（ST），美国博通（broadcom），美国模拟器件（ADI），微控科技（wavecom）。通常这些平台供应商的核心技术都不对外开放，只为购买其开发平台的用户提供一个可二次开发的环境，比如本文所要介绍的自动测试所基于的平台——Agere system， 它在其软件架构的上层为开发用户做了一层 UI（User Interface），并做了最基本的 AL 开发，通常方案提供后可以直接作为国内厂商用于 FTA 测试，这即是国内众多手机厂商和 design house 开发和测试的母体。

曾听一位从事手机功能测试的同仁说“做手机功能测试只要有手就可以了”，确实手机功能测试很容易给人一种是简单而重复按键操作的感觉。但手机功能众多，并且回归测试工作量大，如果单个测试工程师靠手动按键来执行所有测试用例，花费的时间少则几小时，多则需要几天的时间，这样耗费大量测试时间的同时也容易让测试工程师产生疲倦甚至是厌倦心里，很容易造成测试的遗漏。手机测试中常碰到很多重复性高的工作，如发送数条 SMS 或者 MMS 以验证其收发成功率以及稳定性、连续进行多次呼叫、多次对文件系统进行添加删除操作、多任务多进程情况下的冲突测试以及极限测试等等，都是重复性高的工作，手动执行的话费时费力，如果能有一套自动执行的机制，将能大大提高测试的效率。

由于手机平台的特殊性，国内通常都没有自动化测试工具支持手机功能测试，纷繁复杂的功能测试大多只能通过文本化测试用例的指导，由广大测试员手工来完成。手机这种板机的 MMI 功能测试不同于基于 PC 上的 MMI 测试，后者借助 PC 平台，目前市场上已有非常多功能强大且通用的自动测试工具支持其测试，如比较典型的有 Winrunner，Robot，Loadrunner 等等，但这些工具通常不能兼容到象手机这种嵌入式系统中来。当然平台供应商对他们底层 lay1，lay2，lay3 的测试都有自己开发的测试工具来自动执行，但这些工具暂时都不提供给国内的开发厂家。

为了解决上述手机测试工作中的困难，笔者所在的测试团队经过不断的总结实践，目前已在基于杰尔系统（Agere system）平台上建立了一套实用的自动测试机制，通过该机制的建立，不但调动了测试工程师的工作积极性和热情，同时也大大提高了测试的效率。下面将围绕 Agere 平台上自动测试机制给大家做个总体介绍。在讲述该方案前，将先对 Agere 平台的窗体和消息，以及手机同 PC 的数据交互原理做个简单介绍。

二、手机中的窗体和消息：

功能测试时，在手机上每按下一按键，都是在特定的窗口下完成其功能，窗口处理函数接收到窗口所用键盘中定义的按键消息后执行相应的处理，完成指定的工作。这里所谈的窗口系统本质上是一个链表，主要是响应手机中常用的三类消息：用户的按键操作、GSM 网络消息、以及计时器消息。

手机中窗体处理函数结构通常如下：

```
static UINT32  TestWindowProc( UIWINDOW * win, UINT16 cmd, UINT16 wParam, UINT32 lParam )
    {
    switch ( wParam )
            {
                case EV_KEYSEND:            /*按发送键*/
                        CALL(MAOTelNumber);
                        return TRUE;

                case EV_KEYEND:             /*按挂机键*/
                        ENDCALL(MAOTelNumber);
```

```
                return TRUE;

                ◦◦◦◦◦

                default：
                break;
        }
}
```

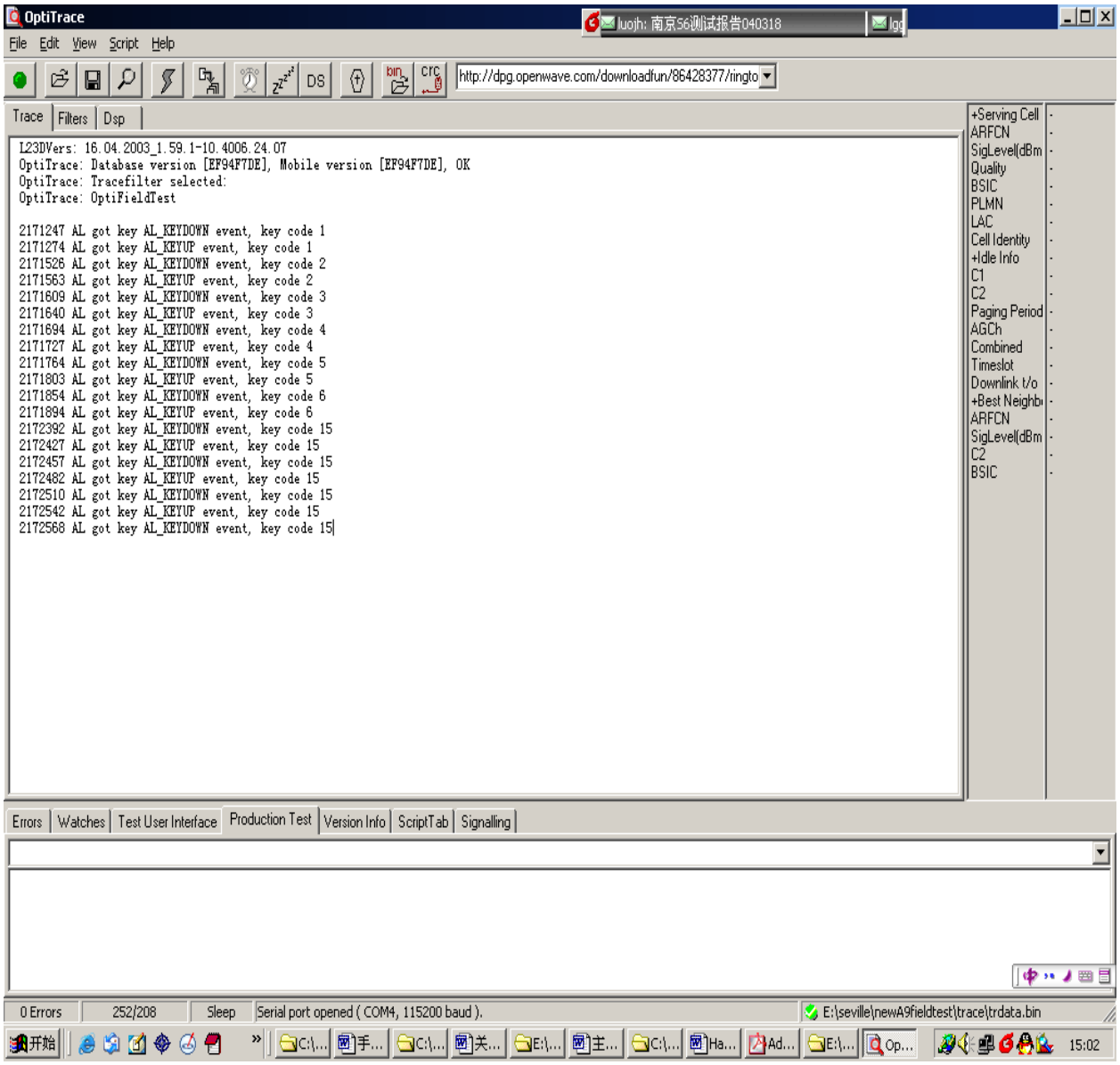
在窗体中除了对消息的实时处理外，还有通过具体的消息传递函数对本窗口中消息进行派发和定向流动，通常有 GSM 消息的流动和键盘消息的流动，派发 GSM 消息时，依据窗口建立的逆向顺序逐层往上流动，而键盘消息只向上传递一层，即子窗口向父窗口传送。 在系统功能测试过程中，窗口中的消息执行情况是看不到摸不着的东西，但是各个窗口中这三类消息的处理以及消息的派发流动都是测试所必须了解和测试的重点，怎样才能直观的看到，跟踪并了解这些消息的执行情况呢？测试工程师可以通过在跟踪点加测试桩或者跟踪语句来实现追踪，利用杰尔系统的 trace 工具（optitrace）以文本的形式输出所需要了解的信息，根据这些信息的输出流程和实际数据，以达到测试跟踪和分析的目的，如上面这一简单例子中所列举的两个事件 EV_KEYSEND 和 EV_KEYEND，最简单的跟踪是通过在这两类事件触发前增加类似于 print 跟踪语句，判断“发送键”按下后是否在指定的窗口里执行到 EV_KEYSEND 事件并调用呼叫函数 CALL 执行呼叫请求,实际运行时，根据 optitrace 工具所显示的 print 信息观察程序的运行及消息的执行情况，跟踪的手段很多，在此就不详细列举。下面介绍 PC 怎样通过 Optitrace 工具实现同手机板机的数据交互。

三、手机与 PC 的数据交互

通常每个平台为软件开发提供一系列的开发套件，常用的有仿真软件、Trace 跟踪分析软件、Download 目标代码的装载软件等等，通过这些软件实现手机同 PC 的数据交互，实现软件的开发仿真，问题的跟踪分析，以及程序的灌写等。这些软件大多采用串口通讯的方式，通过特定的数据线连接手机串口通讯端与 PC 的串口或者 USB 端（USB 转串口）。下面将要介绍的是杰尔系统（Agere system）的开发套件之一 optitrace。

该工具可以运行于 win9X/2000/NT 系统中，是 Agere 参考设计平台的辅助诊断工具，它为软硬件开发人员提供 Protocol Stack and MMI 的跟踪分析以及模拟用户硬件如串口显示和按键，为 field Test 人员提供 Trace Logs 和 Vital signs，为产品测试工程师提供 Product Test environment （PTE） 窗口和脚本的定制以及播放。

该工具的运行界面如下：



以上运行界面中通过 optitrace 工具捕捉的用户按键消息，如 Key Code 4，表示用户在手机上按下数字键 4，key code 后面的数字是按键所定义的编码值，手机中每个按键都有唯一的按键编码值。从中可以看出，用户所有的按键动作都以“AL got key AL_KeyDown event，key code X”的形式被记录下来。这些按键信息的捕捉只是该工具 trace 信息的一部分，该工具提供非常多的 trace 选项，实际应用中，可以

根据所要跟踪的信息来选择显示。

该工具一个最重要的功能是可以在 PC 端通过 PTE 命令模拟用户来操作数据线另外一端的手机，该工具本身定义了 11 类的 PTE 命令，下面重点介绍两个重要的 PTE 命令，

1. 模拟一个按键按下和释放

输入格式：Key　　<INT16 Keycode>

返　　回：Key:DONE

用户可以在 optirace 的 PTE 命令输入行中，通过输入正确的 Key 命令，往手机端写入按键事件，手机端解析后执行相应的按键操作，如用户输入 key 8 回车后，手机端 LCD 显示 8 或者执行按键 8 所对应的操作，执行后返回 key: DONE 消息。同时 trace 中显示 AL got key AL_KeyDown event，key code 8。

2. 定义按键事件的发送间隔

输入格式：Wait　　<INT16　wait time>

返　　回：Key:DONE

举例：

wait 6000　　//等待 6000Ms，即 1 分钟

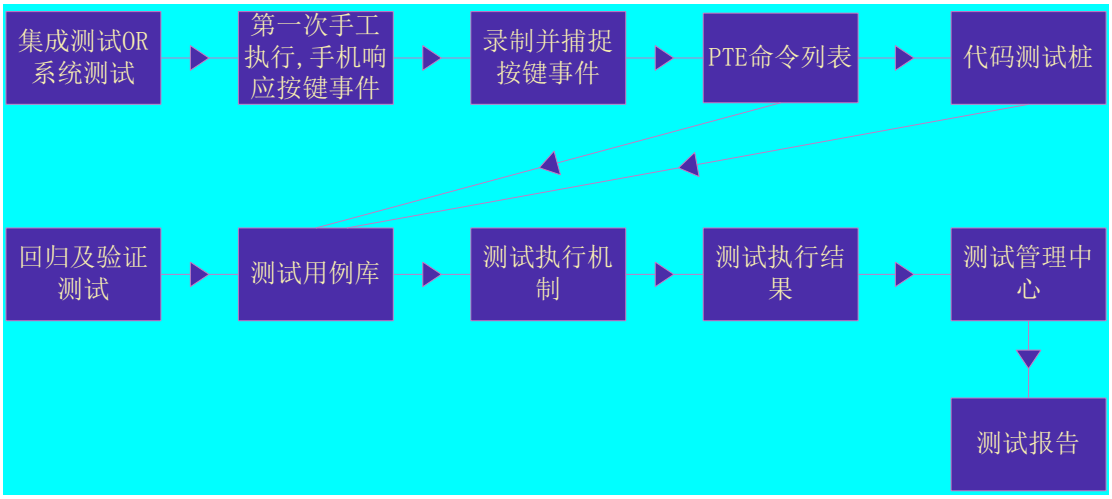
通过该命令，可以请求一个 pause。比如呼叫 1001 通话 1 分钟后挂断。PTE 脚本编写如下：

```
Key 1
Wait 500   //按键间等待 0.5 秒

Key   10
Wait 500

Key 10
Wait 500

Key 1
Wait 500

Key   11      //按呼叫键
Wait   3000   //等待呼叫，3 秒
```

```
Wait    60000    //1001 接通后等待 1 分钟

Key     12       //按挂机键，结束通话
Wait    500
```

四、自动测试方案及框架体系：

下面介绍本公司实践的一套自动化功能测试方案架构，如下图：



1. 方案简述：

自动测试主要工作流程分以下几个主要阶段：

1) 测试用例的设计和准备,形成一套自动测试用例脚本库

自动测试用例的准备，如果贵公司在需求定义的同时有各功能详细具体的 menu tree 架构，那即可在此基础上手动编写 PTE 命令脚本。

如一菜单结构如下：

待机窗口－主菜单－（5）话机设置
- 5－1 关机
- 5－2 闹钟
- 5－3 。。。

假设一手机的关机功能菜单位于主菜单中第 5 项菜单“话机设置”的第一子菜单中，可以用以下脚本方式实现手机执行关机。

```
Key     15    //在待机下按左键进主菜单
Wait 500

Key     5     //按5进入住菜单的第5个子菜单“话机设置”
Wait 500
```

```
Keyhold 1，2000   //长按1键关机
Wait 500
```

从中可以看出只要定义了 menu tree，理解菜单的排列顺序，以及实际的功能操作步骤，即可以用脚本来模拟所有按键和执行步骤来定义测试的 PTE 脚本。

另一种脚本编写方式可以通过录制加转换的方式实现，利用 optitrace 工具录制实际操作时的按键动作，存为 txt 文件，然后将该 txt 文本转换为 PTE 脚本文件。实际测试中通过在集成测试或者系统测试初级阶段录制脚本，这样不会因软件大的变更导致测试用例失效，或者需要大规模维护，降低了风险指数。这些脚本在日后的回归测试中将发挥巨大的作用。

按键录制时测试工程师针对某一功能或者依照某一组测试用例执行一次完整连续的手工测试，通过 optitrace 捕捉本次测试过程中所有的按键事件，生成一份对应的<<按键事件列表文档>>.TXT（optitrace 只能生成文本文档），然后对应将所有按键事件转换为<<*.PTE 文本>>。

2) 代码桩或者跟踪语句

测试时根据实际情况可能需要在各检测点编写用户检验的代码桩或者跟踪语句，代码测试桩有利于对本自动测试体系中软件问题作出较精确的定位和分析，同时也有利于对测试结果的快速判断与自动生成测试报告。这些代码测试桩对应按键事件所对应的程序执行路径和逻辑，主要通过白盒测试方法跟踪代码执行的路径、逻辑覆盖、信息流，数据流和控制流等。在测试执行时，测试桩将执行结果响应并通过 Trace 跟踪语句显示在 optitrace 工具中。编写该测试桩需要测试工程师具备较强的编程能力，同时对手机系统要比较熟悉和了解。各功能完整的代码测试桩的编写工作量非常大，前期可以只针对部分功能的部分特性做尝试。同时测试桩插入在相应的代码中，为了避免混乱，配置时必须将测试代码同程序代码分开，只在测试执行时打开对应的编译开关得到对应的编译版本。

3) 生成一份预期的测试报告

运行预先录制的 PTE 脚本和对应的测试桩，通过 optitrace 工具生成一份预期的测试结果报告(实际就是 optitrace 生成的一份按键事

件和测试桩跟踪输出信息)。这份预期的测试报告日后同实际结果比较，作为实际测试结果与预期结果是否一致的判断。

4) 生成自动测试用例库

最终由<<按键事件列表文档>>、<<*.PTE 文本>>、代码测试桩、<< 预期的测试结果报告>>组成一份自动测试用例。所有的自动测试用例按照一定的结构组织起来形成自动测试用例库。

5) 测试用例的提取并执行

在回归以及后期的验证测试过程中，测试工程师或者程序员对应提取由<<*.PTE 文本>>和测试桩组成的测试用例，执行后生成一份<<实际的测试运行 trace 信息>>，保存该信息，从而测试执行结束。

6) 测试结果分析，生成测试报告

测试结果的分析可以自动和手动执行，手动执行可以通过 Beyond Compare 工具比较<< 预期的测试结果报告>>和<<实际的测试运行 trace 信息>>，即可以得出一份测试的执行报告。

自动生成测试报告比较复杂，需要在 pc 中用高级语言建立一个测试管理中心，该管理中心可用 VC 或者 C++等高级语言编写，在该管理中心中，用户可以选择需要执行的 PTE 脚本或者多个脚本串成的一组脚本，该测试管理中心可以指定测试用例的自动执行，自动提取对应的结果做自动比较分析，从而生成一份对应的测试报告，如果无差异，输出文件中只显示 OK，否则输出差异信息文件。

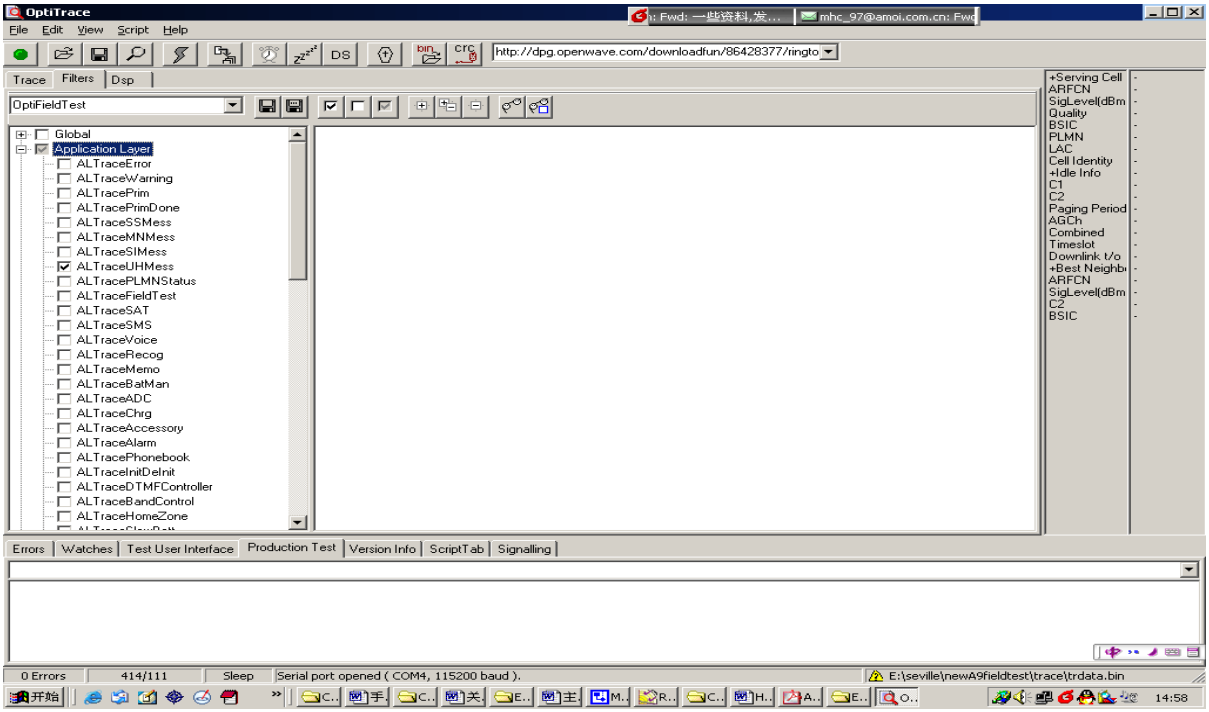
2. 实际应用:

下面以待机下呼叫 1001 共 100 次来测试呼叫成功率的例子来说明上述方案的应用。下面是该例的录制，脚本编写，及实际运行的例子。

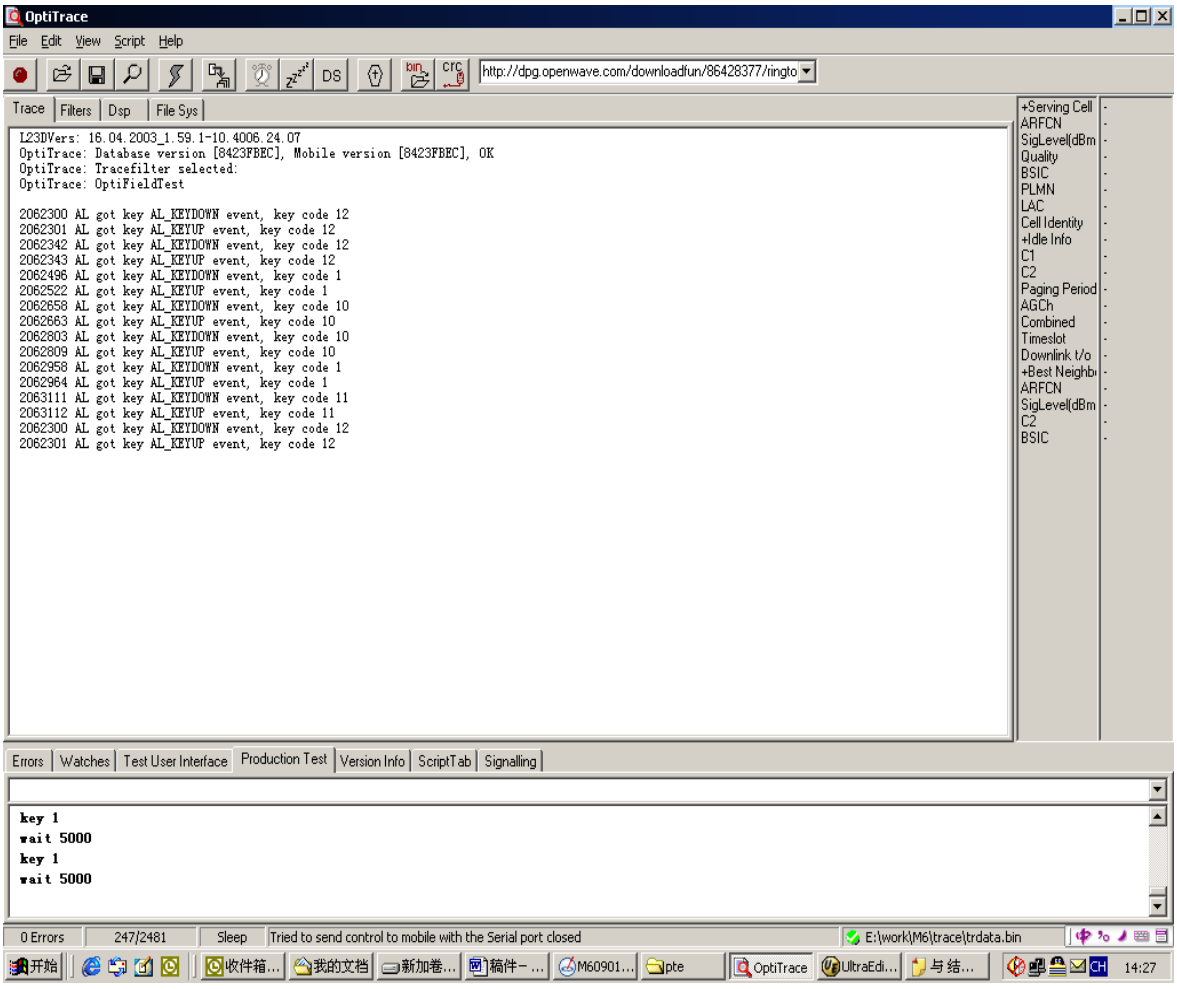
1) 录制按键事件.

首先运行 optitrace.exe 程序

设置 trace 选项,只选择 application layer 中的 ALTraceUHMess 如图所示：



最后手机开机，跑动 trace，测试工程师针对某一功能或者某一组测试用例执行一次完整连续的测试，得到以下按键信息，如图所示。

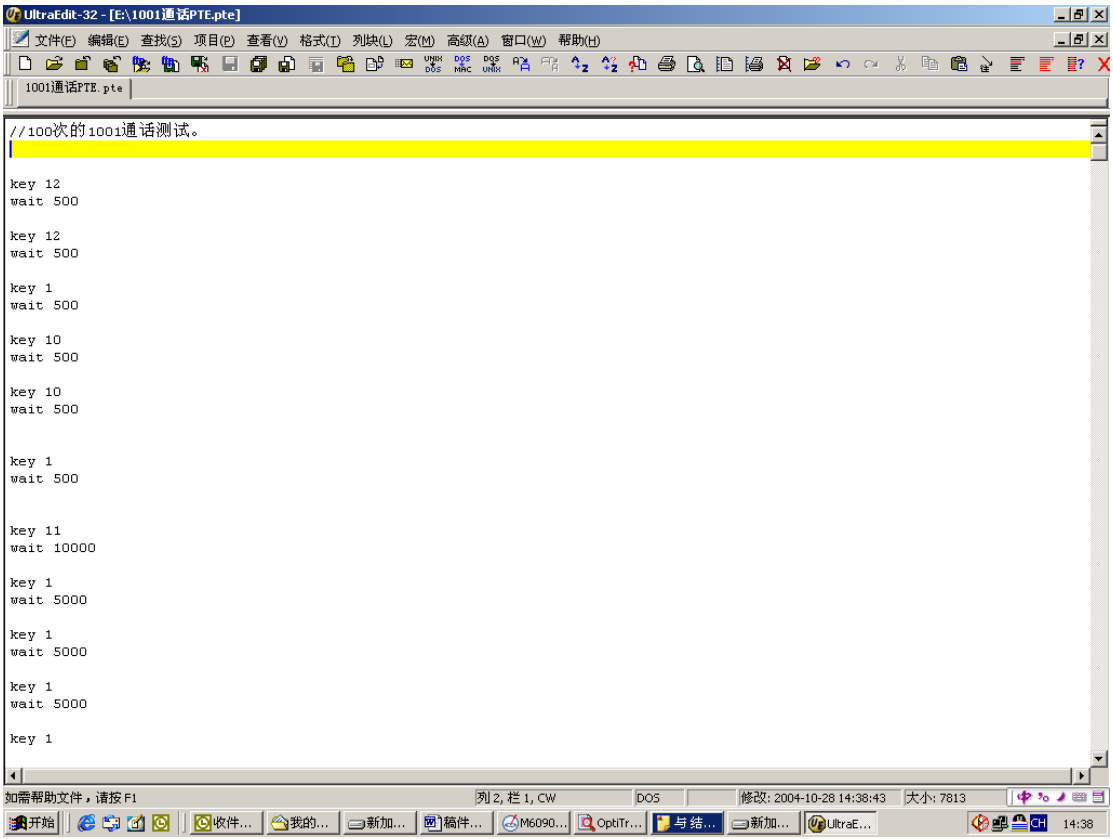


最后测试执行结束后，保存该按键 trace 信息，做好版本记录信息。生成对应事件的按键列表《呼叫 1001 共 100 次.TXT》文档， 该 TXT 文档内容完全同上图所示内容，在次不再重复。

2) 生成 PTE 脚本：

因实际 optitrace 只录制按键消息，需要将这些按键消息转换为 PTE 命令并生成工 optitrace 工具运行的*.PTE 脚本。而通常按键事件众多，手动逐一生成 PTE 脚本非常麻烦，因此需要做一个文件转换工具，逐行提取按键消息转换成 PTE 命令，并做一些相应的注释。

将以上按键列表转换为 PTE 命令列表，生成《呼叫 1001 共 100 次.PTE》文件，转换后的 PTE 脚本如图所示：



3) 编写测试桩：

编写测试代码对需要检测的路径、逻辑覆盖、信息流、数据流和控制流等做测试跟踪，在检测点输出有效的 trace 信息。

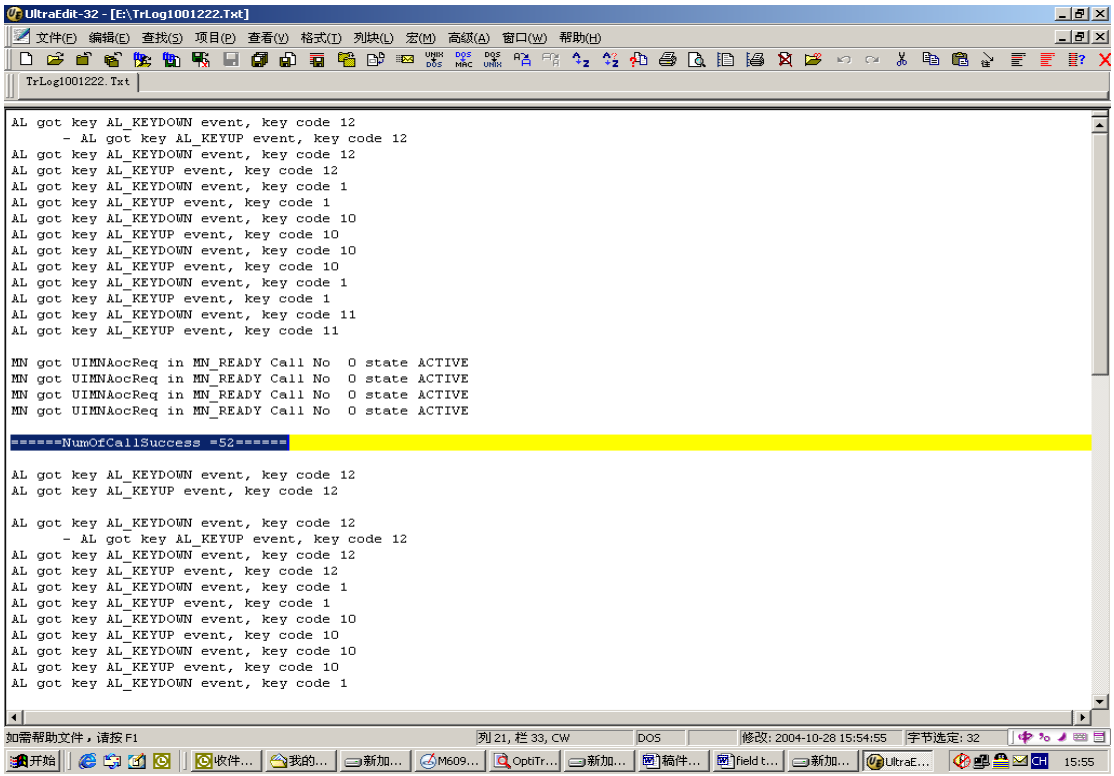
该测试用例比较简单，在此只列举该测试任务中需要关注的呼叫是否成功，记录实际呼叫成功的次数，具体呼叫函数、以及逻辑覆盖因篇幅有限不列举，设计一计数器（NumOfCallSuccess），如果呼叫成功，该计数器累加 1，并且每次呼叫后用 printf 语句在 optitrace 工具上输出该计数器的实际值。

在呼叫窗口的处理函数中，对网络返回的 GSM 消息进行统一处理，在返回的回铃音处理消息中检测呼叫成功即可，如下所示：

```
case    GSMAlerting: //成功接收回铃消息
        if(NumOfCallSuccess < 100)
          GSMprintf("\n====NumOfCallSuccess=%d====
          ==\n",
              ++ NumOfCallSuccess); //呼叫成功
            else
                {
            NumOfCallSuccess =0;
            GSMprintf("\n====== NumOfCallSuccess
                ＝%d======\n", NumOfCallSuccess);
                }
            break;
```

4) 结合以上测试桩，运行《呼叫 1001 共 100 次.PTE》，生成预期的测试结果报告，《呼叫 1001 共 100 次 trace.TXT》的 trace 跟踪记录文件，作为实际测试运行结果比较的依据。

Trace 信息去除无用信息及文件转换后如下所示：



5) 自动运行《呼叫 1001 共 100 次.PTE》，测试结束后目录下共有以下文件：

《呼叫 1001 共 100 次.PTE》：测试运行的脚本

《呼叫 1001 共 100 次 trace.TXT》：预期的测试结果文本，Txt 格式。

《呼叫 1001 共 100 次 trace2.TXT》：实际运行的 trace log 结果，被管理工具转换后的 TXT 文本。

《呼叫 1001 共 100 次.Txt》：测试后生成的测试报告文件，TXT 格式。

五、总结：

本文结合杰尔系统（Agere system）中开发套件 optitrace 工具的使用，从 PTE 脚本的制作，到回归测试中脚本的测试运行，介绍了一个测试团队在手机功能级测试中采用的自动化方案，本团队在实际的使用中感受了该自动化测试方案所带来的乐趣和效率，在此著成本文供大家一起探讨，最后感谢本文的所有读者，如果您能从中汲取一点有用的营养，得到一些帮助，那我将感到无限的欣慰，这也是我整理这篇手机自动测试资料的初衷。

由于时间仓促水平有限，错误之处在所难免，敬请广大读者批评指正。

ISO 和 CMM，我们该选择谁呢？

秋　南

【摘要】本文抛开以往比较 CMM 和 ISO 两种质量管理体系时仅从体系本身的特点上入手的套路，尝试从两种体系在组织实施过程中在“管理水平适用性、组织机构复杂度和研发过程复杂度的适用性、量化管理的适用性”上入手，比较并阐明组织如何恰当的选用本组织本阶段优选的质量管理体系。

【关键词】软件成熟度模型、ISO 质量管理体系、组织机构复杂度、研发过程复杂度

网上评价 ISO 和 CMM 差异和共同点的文章比较多，但大多是从标准条文本身进行比对的。下面我将从适用性方面对比一下 ISO 和 CMM（I）两套质量管理体系框架。

一、管理水平的适用性

ISO 系列标准虽是就质量管理而言，然而也可以看作一个浓缩的管理学框架，尤其是作为认证审核依据的 ISO 9004：2000。它是一个大而全的系统，几乎覆盖了公司管理的各个方面。它对一个尚无完整质量管理体系、依靠自身发展过程中自发、被动或应急性的制定一些组织管理和研发管理过程和规范的公司来说，是一个很好的快速建立公司系统化管理框架的参照体系。通过这个体系的建立，把公司的各组织机构的业务流程、接口关系、人员岗位及部门职能界定、各种公司管理制度有机的整合起来。如果在这个过程中能够配合 BPR 进行，彻底梳理原先大家都意识到、但却无从下手的管理沉疴，效果会更好。

CMM 是对软件开发实践所涉及的整个工程流程的规定和分析，它的体系既包括软件工程过程本身，也包括对这一过程的管理。它更多的是提示我们所处各个管理成熟度等级的阶段目标，以及为了达到这些特定的阶段目标而分解的大量具体实践。它没有告诉我们怎么达到这个目标，因此它是基于组织（公司）能意识到自身差距并能针对性的采取改进措施。它更多的是站在各个具体的点上去思考问题。纵观 CMM2 的各个 KPA，如果公司连保证这些 KPA 实施的管理基础都不具备，从二级 KPA 的一个个点上去突破，很难达到以点带面的效果。

这是一种比较可取的方式是先来一场组织管理上的革命，去创造这种质量管理水平提高的管理基础。对于一个具有一定管理基础的组织，引入 CMM（I），从这些具体的点上去采取措施，对于组织过程能力的持续提高、组织过程质量的持续改进将起到极大的推动作用。而组织管理革命的一个利器正是 ISO9000 质量管理体系。

当然，我们在组织建立 ISO9000 质量管理体系的过程中，也可以借鉴 CMM 的思想，抓阶段的关键问题，分步实施，逐步推进。因为面对 ISO9000 这一标准，对大多数组织来说，要一下子全面执行是有很大难度的，并且其有效性也难于体现出来。这就需要组织根据自身的实际情况，分析组织的现状、资源、文化、人员素质和面临的社会环境，找出同 ISO 9004 标准的差距，制订一个中长期的目标计划，从选择一、二个 ISO 9004 的条款要求的试点做起，成熟一项巩固一项，逐个地推广应用，并制订出具体的实施进度计划，在一定的期限内完成 ISO 9004 标准全面贯彻执行工作。

由此，我更倾向于一个公司先基于 ISO9000 质量管理体系建立起质量管理体系框架，培养组织的质量意识。然后在此基础上，选择若干过程域进行重点监控，以逐步达到 CMM 某个成熟度等级的要求。

二、复杂度的适用性

这里所讲的复杂度包含两个方面的意思：一是指研发过程本身的复杂度；另一是指组织机构的复杂度。

基于我对 ISO 和 CMM 以下理解：ISO 是一个达标性的体系，大而全，比较偏向于宏观把握；CMM 则着眼于具体实践，对各个关键过程域进行分解细化，偏向于微观控制。

个人倾向于在一个研发过程本身复杂度不是很高，但组织复杂度由于组织本身的发展变得庞杂的组织中推行基于 ISO 的质量管理体系；在一个组织机构相对完善，由于项目规模本身的扩大而迫切需要引入适当的过程监控的组织实施基于 CMM 的质量管理体系。

下面阐述一下两个概念：

何谓研发过程复杂度？

本文定义的研发过程复杂度是指针对组织正在实施的单个项目（或产品，以下统称项目）研发活动而言的。主要考虑项目规模（项目范围和工作产品规模）、项目团队规模、项目周期等因素。本文考虑最核心的指标为项目团队规模。

何谓组织机构复杂度？

本文定义的组织机构复杂度是指组织的人员规模、组织的机构设置层次结构及接口关系上。

一个组织结构复杂度高的公司，并不意味着它的研发过程复杂度高。但一个研发过程复杂度较高的组织，往往可以推断其组织结构复杂度较高。如：一个 1000 多人的从事超市、酒店行业 ERP 的软件公司、其人员分布于各 30 人月以内（6 人 5 月左右）的各小项目的实施，公司按照地域和业务领域划分 4 个事业部，分四级进行管理（总裁—>事业部总经理—>部门经理—>员工）。按照本文定义的复杂度含义可以断言，该公司的组织结构复杂度较高（四层管理、1000 人以上规模），但其研发过程复杂度较低（单个项目投入人数在 10 人内）。

为了更好的对比分析请看下面的统计数据：

到 1999 年止，全球范围内共进行了 1330 次 CMM 评测，总计评测项目有 5452 项，参加评测的机构逐年攀升，其中有 7.2%是海外项目，参加国别有 34 个，参加机构类型，商业机构占 56.1%，美国国防部供应商占 29.8%，军方和政府机构占 10.5%。其中，初始级机构占总评估数的 43.2%，可重复级占 34.2%，定义级占 17.3%，管理级占 4%，优化级占 1.4%。第二级（可重复级）比例最高的为 25 人到 100 人的机构，第三级（定义级）所占比例最高的为 1000 到 2000 人的企业，第五级（优化级）所占比例最高的为 2000 人以上的企业。

于是乎国内各咨询机构和各有意使企业通过认证的管理者得出结论，国内的企业可以以上述统计数据为参照，选择适合自己的认证目标。各位看官请试想一下：一个由众多独立核算的事业部或子公司粘合而成的一个 1000 多人的公司，分各个业务方向从事着众多 30 人月（3－5 个人，6－10 个月）上下规模的项目，其研发过程复杂度显然是不高的，其研发过程的可视性相对来说比较容易达到。此时组织的主要焦点在于如何降低其组织的管理复杂度，提高管理过程的可视性。如果此时严格照搬适合大团队研发过程管理的过程管理体系，到头来肯定是过程监控与研发实施活动成为脱节的两张皮，为了达到体系的要求去补大量的记录，为了真正不受羁绊的去更好的实施项目跨过很多死板过程的羁绊。个人倾向类似情况的组织（公司）真正从整理业务流程入手，建立一套着眼点在提高组织管理过程可视性的管理体系，不妨老老实实的按照 ISO 体系的要求，建立自己的质量管理体系。

对应的例子是一个 100 多人的公司，其 100 多人的开发团队投入

其核心产品的开发，所有组织活动都是围绕这个产品开发过程展开的。此时组织的质量瓶颈在于如何提高研发过程的可视性。此类组织我倾向于采用 CMM 建立逐步完善的研发过程质量监控体系。

综上所述，从复杂度与质量管理体系选择上，可以得到下表的对应关系：

组织机构复杂度	研发过程复杂度	关键问题	选择体系
高	高	管理过程可视性 研发过程可视性	ISO CMM
高	低	管理过程可视性	ISO
低	高	研发过程可视性	CMM
低	低		

三、量化管理的适用性上

CMM 和 ISO 9000 都强调过程改进。如果组织还没有一个文档化的研发过程，则首要任务是对当前的研发过程进行抽象、分析、整理并文档化，从而制定出一个符合本组织研发实际的研发过程规范，并从制度上确保研发过程规范的执行。

如果已经具备了相对陈述的研发过程规范，组织需要做的是对这个研发过程规范在实际研发活动中的表现进行持续的评估，找出问题，然后对过程规范本身进行补充修订，然后找项目试点，最终推广应用。这也就是我们通常所说的持续改进。

在阐述量化管理的适用性前，先来探讨一下为什么要进行量化管理？

1、 随着组织的质量管理水平和质量意识的提高，越来越多的过程质量问题的不是可以通过直接的外在表现可以观察出来的。

2、 随着组织的质量管理水平和质量意识的提高，越来越多的过程质量问题的定位越来越难以通过其直接的外在表现进行判断。

如何发现过程质量问题？如何定位过程质量问题的根源所在？CMM L4 的“定量过程管理”、CMMI L3 的“决策分析和决定”和 CMMI L2 的“测量和分析”KPA 很好的回答了这个问题：我们通过收集、整理和分析过程数据，并运用系统的决策方法从中得出分析结论，寻求问题的解决方法。

请看下例：

某组织最高管理者 2004 年以来，接到客户持续不断的抱怨，说系统稳定性差？当最高管理者在向产品线反馈该问题时，产品线说，我们的产品越来越稳定，去年的外部故障是 15 个，今年的外部故障只有 13 个。一切都那么的 OK，为啥客户的抱怨却与日俱增呢？问题到底出现在哪里？该如何去纠正？

产品线成立了专门小组寻找问题所在及解决之道。在收集数据的过程中，他们关注到这样一个现象，2003 年产品售出 256 套（25 个客户），2004 年度产品一下子售出 6000 套（300 个客户）。虽然报出的外部故障只有 13 个，比去年的 15 个有了减少，但其波及的范围（客户数）却成 10 倍的增加。

问题症结终于找到了，我们产品质量的提高没有跟上我们市场扩张的速度。我们需要解决的最关键的问题是彻底解决现有故障，并作充分的测试，提高我们发布产品的稳定性。

过程管理的核心是使过程状态可见并使过程可控；量化管理的目标是如何使前者更加精准。ISO9000 虽然也强调以事实为依据的科学决策和 SPC，并要求组织有这方面文档化的指南与要求，但它没有进一步说明从哪些关键点上去入手，而且它的 SPC 更多的是针对制造业的不良品而考虑的。

CMM/CMMI-SW 是专门为软件研发过程管理设计的，它在各相关 KPA/PA 中蕴涵的量化管理的关键实践更具针对性、更细致、也更全面。CMMI L4 更是明确要求：“对各个过程运用统计技术和其他定量技术对各个过程实施控制，建立了关于产品质量、服务质量以及过程性能的定量目标，并且把这些定量目标作为管理过程的准则。在过程的整个生存周期中，对产品质量、服务质量和过程性能都进行统计管理。”在第 4 级上，对过程的性能是以统计技术或其他定量技术进行控制。并且从统计意义上说是可预见的。

可见，在具有一定质量管理基础的组织内，可以参照 CMM/SMMI-SW 相关 KPA/PA 的要求，建立自己的研发过程度量体系，逐步推进量化管理。

结论：

组织在建立质量管理体系时，可结合本组织的管理水平、组织机构复杂度、研发过程复杂度的现状综合考虑，选择适合本组织的模型和体系。并可以在过程中参照 CMM/CMM-SW/SE 的要求，提高组织

的量化管理水平。

参考文献：

1、《能力成熟度模型（CMM）：软件过程改进指南》 电子工业出版社 刘孟仁 等译 [美] 卡耐基梅隆大学 2001 年

2、部分数据及案例来自互联网，未能一一列示，敬请谅解

性能测试工具之研究

王玉亭

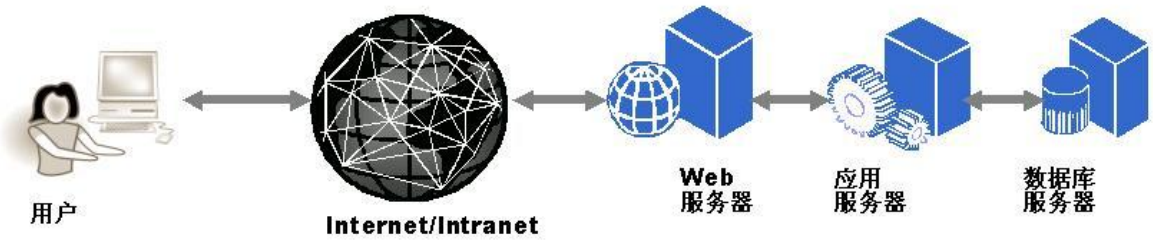
【摘要】本文在分析市场上已有的商用的性能测试工具实现原理和体系架构的基础上，提出了利用现有的开源代码构建开源的性能测试工具思路。

【关键词】性能测试、Vugen、Conductor、Player、Analysis

一、性能测试的意义

追求更高的质量和更高的性能是人类的天性。“更高，更快，更强”的奥运会是对人类自身运动能力的测试。同样，人类也在追求我们工作生活中不可或缺的 IT 系统能够提供更快更强的服务。目前 IT 系统已经称为各个企业运转业务时最重要的系统之一。对 IT 历史稍微有所了解的人都知道，IT 系统经过早年的一个人使用的单机系统时代，几十个人使用的局域网中的客户机服务器系统时代，到现在服务成千上万用户的跨广域网的庞大系统时代。IT 系统发展中的最明显的特征之一就是所谓的“数据大集中”，即数据越来越集中到后台的服务器中，系统同时为成百上千，乃至上万的用户提供服务。这样的例子在银行、保险、电信公司中随处可见。随着企业业务量的加大，其 IT 系统承载的负荷越来越重，系统性能的好坏严重影响了企业对外提供服务的质量。对 IT 系统的性能进行测试和调优越来越引起企业的重视。

目前，典型的企业 IT 系统的架构如下所示：



这样的系统由客户端、网络、防火墙、负载均衡器、Web 服务器、应用服务器(中间件)、数据库等等环节组成，根据木桶原理，即木桶所能装的水的量取决于最短的那块木板，整个系统的性能要得到提

高，每个环节的性能都需要优化。在这样的 IT 系统中，每个环节的都是一个很复杂的子系统，对其调优都是一门专门的技能。Oracle 数据库的调优就需要专门的技能和多年的经验。对于整个 IT 系统的调优，其复杂程度更是急剧增加。因此 IT 系统性能测试调优是一个复杂的项目，需要拥有各种专门技能的专家组成小组来完成。这些专家包括操作系统专家、网络专家、数据库专家、应用服务器专家、应用软件和业务专家等等。

虽然性能测试是一项很复杂和专业的工作，但是由于企业 IT 系统的重要性，保证其性能的稳定对于企业对外提供优质服务越来越得到企业的重视。性能测试服务的市场正在快速发育中。研究系统性能测试越来越有意义。

要保证性能测试项目的高质量，必需依赖两个重要的因素：人和工具。具有多年经验的高素质的专家小组是保证性能测试的最重要的因素。另一方面，功能全面、使用灵活的性能测试工具对于加速性能测试，提高测试质量和效果也是必不可少的。

现在有很多种性能测试工具，从功能简单单一的开放源码的软件到昂贵的商业性能测试工具。本文论述了性能测试工具的一般体系架构和技术要点，并探讨了利用已经存在的开放源码的软件整合出一套开源的优质的性能测试工具的可行性。

二、性能测试工具综述

性能测试的主要手段是通过产生模拟真实业务的压力对被测系统进行加压，研究被测系统在不同压力情况下的表现，找出其潜在的瓶颈。因此，一个良好的性能测试工具必需能做到以下几点：

- 提供产生压力的手段
- 能够对后台系统进行监控
- 对压力数据能够进行分析，快速找出被测系统的瓶颈。

产生压力的手段，主要是通过编写压力脚本，这些脚本以多个进程或者线程的形式在客户端进行运行，来模拟多用户对被测系统的并发访问，以此达到产生压力的目的。由于一台普通的 PC 机可以轻易产生几百乃至上千个进程或线程，通过使用若干台 PC 机，就可以轻易模拟出成千上万个并发用户。压力脚本执行的功能和被测系统客户端软件执行的功能应该一样，从而产生真正的业务压力。 编写压力

脚本的工作实际上就是重新编写客户端软件。为了快速达到编写脚本的目的，采用的最有效的方式是通过性能测试工具录制客户端软件和服务器之间的通讯包，自动产生脚本，然后在自动生成的脚本的基础上进行少量修改，如：关联动态内容，指定批量测试数据等。根据经验，压力脚本的准备工作往往占据整个性能测试项目的 50%的时间和工作量。 因此，能否提供录制和自动产生脚本的功能是性能测试工具最主要的评价指标之一。

压力脚本的方式给我们提供了模拟各种压力状况的有力手段，通过人为制造各种类型的压力，我们可以观察被测系统在各种压力状况下的表现，从而定位系统瓶颈，作为系统调优的基础。因此，提供丰富的对后台系统进行监控的手段是性能测试工具第二个最主要的评价指标。监控应该不在被测系统上安装任何软件，否则的话，为了监控而引入的“代理”之类的小软件将给被测系统引入新的可变因素，一方面造成了测试结果的不准确，另一方面会给用户的系统的稳定性可能造成影响，从而导致用户的反感和拒绝。目前，各种监控手段大都采用所谓“无代理”的方式，即不在被测系统上安装任何软件，仅仅通过改变被测系统的配置，就可以对被测系统进行监控。需要监控的部件多种多样，包括操作系统、数据库、中间件、应用系统、安全模块、网络、防火墙等等。

一组压力测试运行完毕后，我们会得到详尽的性能数据。这些数据包括最终用户的响应时间，后台系统各个部件的运行数据。这些数据的量非常大，往往包括几千个变量的运行曲线，大小可能达到上 G 的规模。靠人工去分析这些数据几乎是不可能的，性能测试工具必需提供数据分析工具，帮助性能测试人员去阅读、解读和分析数据，辅助测试人员定位系统的瓶颈。数据分析工具是保证最终测试成果的手段，因此它是性能测试工具中最重要的部分之一。

目前已经存在的性能测试工具林林总总，数量不下一百种，从单一的开放源码的免费小工具如 Aapache 自带的 web 性能测试工具 ab 到大而全的商业性能测试软件如 Mercury 的 LoadRunner 等等。任何性能测试工具都有其优缺点，我们可以根据实际情况挑选用最合适的工具。

前面已经指出，性能测试是一项复杂的工作，一个性能测试项目的质量如何，测试人员的素质、能力和经验是最关键的因素。拥有世界上最先进的 CT 扫描仪并不能让你成为一个优秀的医生。不过，“工欲善其事，必先利其器”， 拥有一套自己非常熟悉，功能全面、质量

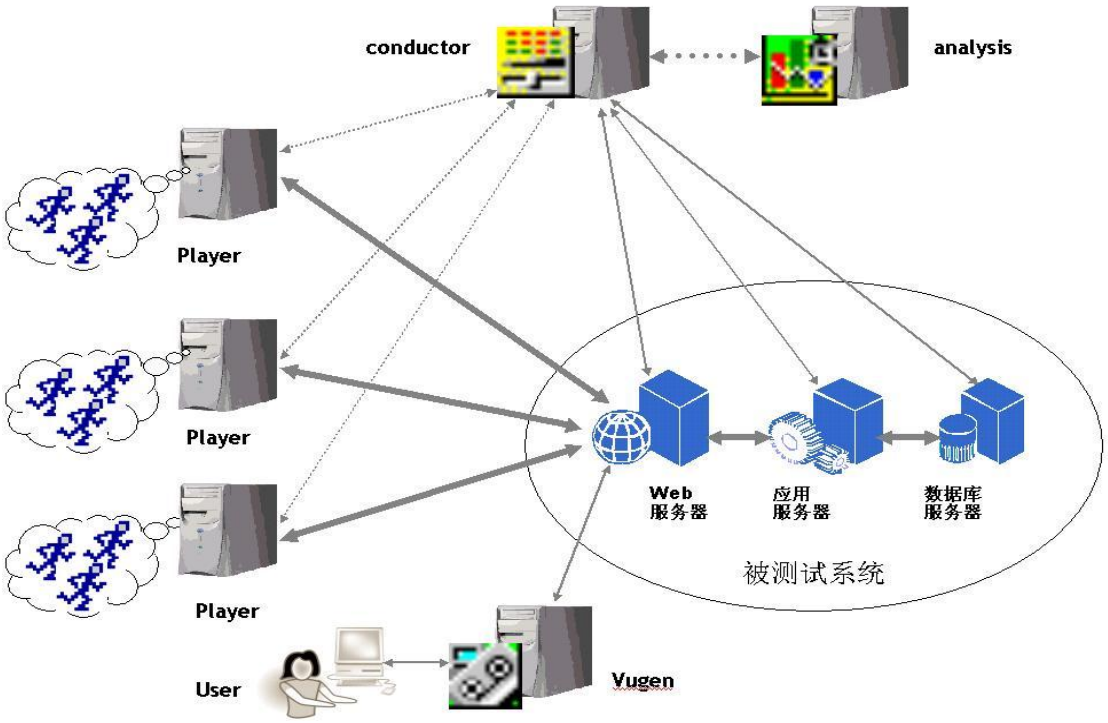
可靠的性能测试工具对于从事性能测试的人员非常有吸引力。在商业性能测试软件中，大多价格非常昂贵。由于大多数性能测试工具是按照并发用户数收取费用，因此要获得大的并发量的价格是很高的。虽然存在很多免费的性能测试工具，但其功能不足，彼此不成系统，不能灵活搭配使用。

一套功能全面的性能测试工具的开发工作量是非常大的，这也是为什么商业性能测试软件价格昂贵的主要因素之一。由于互联网和开放源码运动的发展，性能测试工具的各种功能都以各种形式的开源软件存在了。**如果我们设计出一套合理的架构，在统一的架构下整合各种缺乏系统性的开源工具软件，使之能够彼此配套，搭配出一套功能全面、质量可靠，而且是开放源码的性能测试工具是完全有可能的。**

本文的下面部分具体论述性能测试工具的基本框架和技术要点，希望热爱编程，希望对开放源码运动有所贡献的读者能从本文的论述中获得一些启发，沿着作者的思路继续往前行。

三、性能测试工具的体系架构

作者对性能测试工具 LoadRunner 比较熟悉，通过对 LoadRunner 的了解和评估，作者设计的性能测试工具体系架构如下图所示：



性能测试工具的组成部分有如下几个：

- 虚拟用户脚本产生器 Vugen(Virtual User Generator)

- 压力调度和监控系统 Conductor
- 压力产生器 Player
- 压力结果分析工具 Analysis

通常，进行性能测试项目的一般步骤如下：

1. 用户确定需要录制的交易，通过用户操作和 Vugen 的录制，记录并生成自动化脚本。
2. 修改脚本，确定脚本能够回放成功。
3. Conductor 是一个集中控制平台，它和压力产生器 player 互连，指定脚本在 player 上的分配，并控制 player 向被测系统的加压方式和行为。
4. Conductor 同时负责搜集被测系统的各个环节的性能数据。各个 Player 会记录最终用户响应时间和脚本执行的日志。
5. 压力运行结束以后，Player 将数据传送到 Conductor 中，Conductor 负责将数据汇总。
6. 数据分析工具 Analysis 读取压力测试数据，进行分析工作，确定瓶颈和调优方法。
7. 针对性地进行系统调优，重复进行压力测试，确定性能是否得到提高。
8. 重复以上 3-7 步，逐步提高系统的性能。

鲁制脚本的工具虚拟用户产生器 Vugen 实际上是一套开发调试工具。Conductor 是一个框架程序和监控程序，它负责将 Vugen 开发的脚本以多进程/多线程的方式在 Player 机器上运行。为了产生更大的压力，Conductor 必需支持集群功能，理论上 Conductor 可以和任意多台 Player 机器互连，以便产生足够大的负载压力。Conductor 同时实现无代理方式的监控功能，可以监控各种主流的软件，并且提供对不支持的软件进行监控的二次开发的手段。Analysis 实际上是一个数据分析工具，用于事后的数据分析，它可以安装在任何 Windows 平台的机器上。下面我们论述每个部件的技术要点。

四、虚拟用户产生器 Vugen

虚拟用户产生器通过录制客户端和后台服务器之间的通讯包，分析其中的协议，自动产生脚本。用户在自动产生的脚本的基础上进

行修改，从而快速开发出一个逻辑功能和客户端软件完全一样的压力脚本程序。

录制的技术主要是通过 proxy 的方式来实现的，如下图所示：

Vugen 根据对捕获的数据的分析，将其还原成对应协议的 API 组成的脚本。由于 Proxy 源程序的获得非常容易，Vugen 的主要的技术要点是**如何根据捕获的数据包来反解析成对应的网络协议**。通常捕获的数据包为 TCP 数据流，我们可以很容易的生成 socket 层次的脚本，类似如下示例：

```
int main( int argc, char** argv)
{
char buf[BUF_MAX_LEN];
int socket = 0;

    socket = connect("IP=192.168.52.65", "Port=3200", TCP);
    getbuffer(buf, "trace.dat", 1, SEND);
    send(socket, buf);
    receive(socket, buf);

    getbuffer(buf, "trace.dat", 3, SEND);
    send(socket, buf);
    receive(socket, buf);

    ……
    close(socket);
}
```

其中 trace.dat 包含着录制时捕获的数据包，按照“发-收-发-收-发-收”的顺序排放。

毫无疑问，这样的脚本按照记录的收发过程来回放，但是它的最大的缺点是处于太底层。要分析和修改 socket API 的脚本以及数据包

的具体内容将是一个繁重而且烦人的工作，进行关联的工作的难度也将大大提高。客户端程序往往是利用更高层的应用协议 API 编写的，客户端软件的编写者也不一定对 socket-API 组成脚本进行修改。因此，**Vugen 应该尽可能地产生更高层网络协议的脚本**，方便用户的阅读和修改工作。

以 Tuxedo 应用为例，对于 Tuxedo 应用，一次典型的 Tuxedo 业务调用的序列为：

```
tpinit(....)   //和后台建立连接
tpcall(....)   //向后台发起交易请求
tpterm(...)   //  断开和后台的连接
```

这三次 api 调用将产生 TCP 层的 7 次收发动作，Vugen 必需根据这 7 次的收发，还原产生 Tuxedo-API 的调用序列。

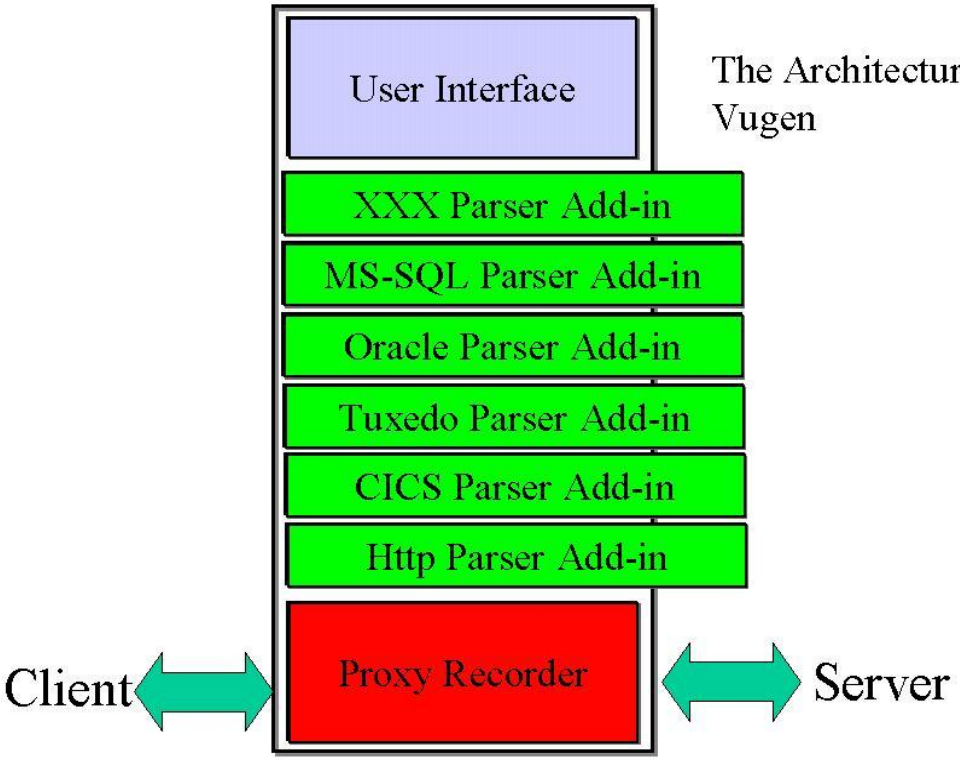
由于对于不同的应用层协议，只能分别开发，因此，Vugen 支持的网络协议的多少是衡量性能测试工具的主要指标之一。

当然，socket 方式是一切应用层协议的基础，socket 脚本是一种通用的方式。对于 Vugen 不支持的应用层协议只能通过 socket 层次来录制。因此 Vugen 能生成 socket-API 脚本是其最基本的功能。

VuGen 产生的脚本应该应该是跨平台的，因此它应该提供 C/Java 两种语言的方式，支持各种平台的 C/Java 编译器。脚本可以在 Windows/Unix/Linux 上运行，Player 运行的机器既可以是 Windows 平台，也可以是 Linux, FreeBSD, Solaris, AIX 和 HP-UX 等平台，这样会方便用户选择机器作为 Player。这一点非常重要。

由于 Vugen 支持的每种应用层协议必需单独开发，在设计 VuGen 的软件体系架构时，应该采用插件的方式来设计网络协议的解析器。这部分的设计借鉴了 Apache 的 module 设计思想，可以让任何对开发协议解析器感兴趣的开发者在一个统一的框架下开发。

VuGen 的体系结构如下图所示：



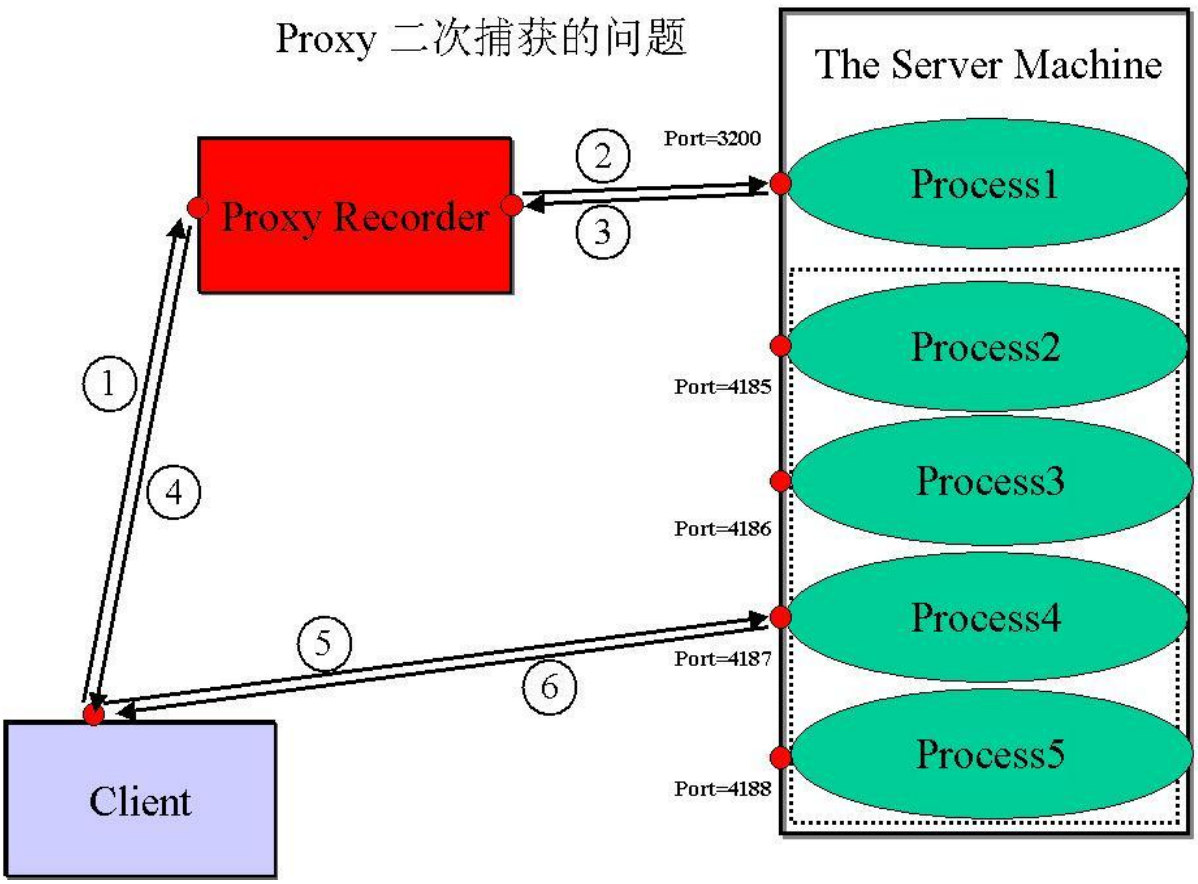
Vugen 的体系结构分为三部分：

- 第一部分为底层 proxy 录制器，负责捕获客户端和服务器之间通讯的数据包，这样的软件在开放源码的世界里面随处可见，而且非常成熟。 我们只要移植过来就可以使用。
- 第二部分是界面部分，提供脚本编辑、调试和运行功能，这部分可以用 Visual C++/MFC 实现 Windows 平台版本和 Java/AWT 实现 Unix 版本。
- 第三部分是以插件的形式提供的分析各种网络协议的解析器。开发这类插件的强有力的开发工具为 lex 和 yacc。

五、Proxy 二次捕获的问题

Vugen 的 Proxy 方式需要解决的一个问题是二次捕获数据包的问题。

早期的网络服务器程序对外提供一个固定端口，客户端仅仅和这个端口通讯就可以了。这对于 proxy 录制非常容易。但是现在很多服务器程序为了提高对客户端并发量的支持，采用两个端口通讯的方式。如下图所示：



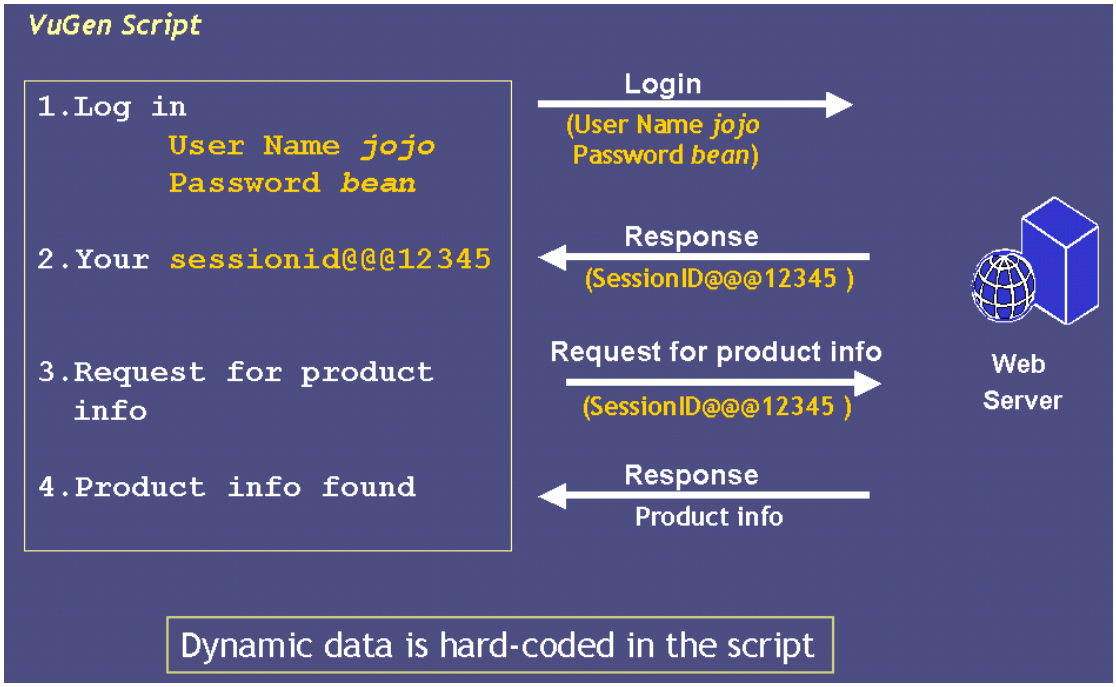
整个通讯的过程如下：

- 第一步：客户端将请求发往 Proxy 录制器。
- 第二步：Proxy 录制器将请求发往真正的服务器的指定端口，即图中的 3200 端口。
- 第三步:服务器机器的 3200 端口返回数据包给 Proxy 录制器。该数据包中包含了下一次通讯的目的地址，形式为 IP:Port。很显然，这里的 IP 数据为真正服务器的 IP 地址。
- 第四步，Proxy 录制器把请求返回给客户端。
- 第五步，客户端根据提供的 IP:Port 信息直接把请求发往真正的服务器，不再经过 Proxy 录制器。
- 第六步：以后的通讯只是客户端和服务器之间的通讯了，Proxy 录制器是无法捕获这些通讯包了。

因此一般的 Proxy 录制器只能捕获头两个收发的数据包。 这个问题更一般的情形的例子是 HTTP 的 redirection 功能。第二次通讯可能发往另外一台机器了。Proxy 录制器必需解决这个问题。

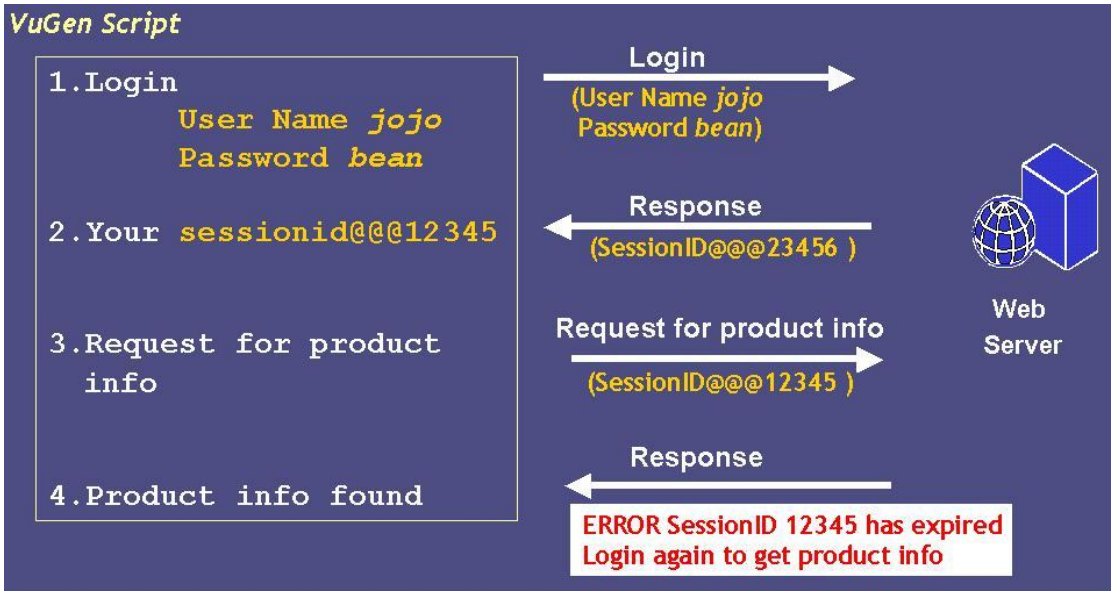
六、关联的问题

客户端和服务器之间的通讯，有一部分是数据是动态的，每次通讯都不一样。Proxy 录制器在录制的时候是无法区分哪些是静态的信息，哪些是动态的信息，所有的信息都以 hard-coded 的方式记录下来。但是在回放的时候，如果有些信息不改变，那么脚本是不能执行成功的。考虑如下情形：



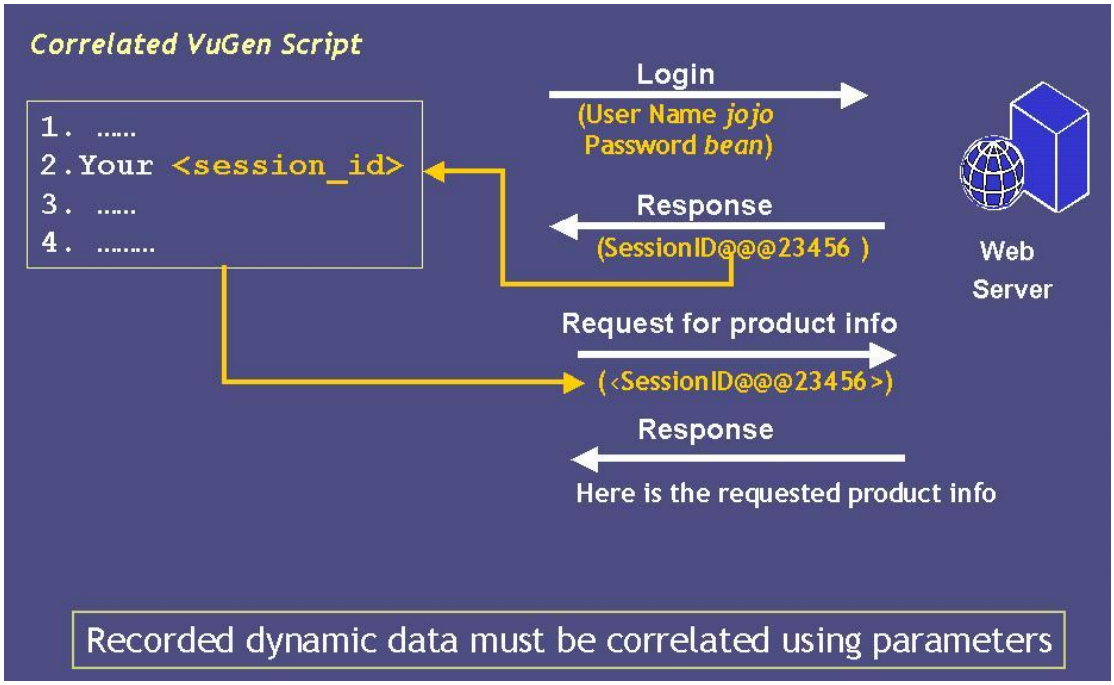
如上图所示，用户 jojo 以 jojo/bean 的账号/口令登录某一 web 服务器，查询某产品的信息，由 Vugen 录制交易的全部通讯包。web 服务器返回给 jojo 一个动态的会话 ID: SessionID@12345，作为这次登录的会话标识。由于 Vugen 无法知道哪些信息是动态的，它会照单全收的方式，把所有的数据就记录下来。接着 jojo 根据 Web 服务器告诉他的 SessionID 去查询产品列表，交易可以正常执行下去。

我们会观察到，当 Vugen 根据捕获的通讯包生成 http 脚本的时候，SessionID 是 Hard-coded 的，即“写死”在程序里面的。当我们不加修改的回放该脚本的时候，会出现什么问题呢？如下图所示：



按照录制时候的脚本，jojo 以 jojo/bean 登录后，Web 服务器返回给 jojo 一个动态会话 ID: SessionID@23456，这个值已经不是录制时候产生的 SessionID@12345 了，而是新的值：SessionID@23456。那么脚本根据记录的 SessionID 的值，仍然会使用 SessionID@12345 去执行下面的查询交易。由于会话 ID 是有时效性的，用户退出系统后，其 SessionID 会失效，那么，服务器会给出一个“SessionID 失效”的错误，从而导致脚本无法正常执行下去。

对于上面的问题的通用解决方法如下图所示：



在第一次从服务器得到 SessionID 的时候把其放在一个变量<session_id>里面，在后面脚本访问服务器的语句里面，把所有

的”SessionID@12345”替换为变量<session_id>就可以圆满解决这个问题。

这种问题在任何系统都是非常常见对外问题。其通用的模式是：“服务器返回给客户端一些动态变化的值，客户端使用这些值去访问服务器的时候，不能把这些值写死在脚本里面，而应该存放在一个变量里面。” 这就是关联的概念。

关联的工作往往占据开发脚本的大部分时间，因为我们必需针对每一个具体的系统进行细致的分析，确定其需要关联的动态信息。为了快速开发脚本，Vugen 必需提供帮助我们关联的手段，最好做到自动关联。自动关联的方法有三种：

- 在录制之前设定辨别规则，录制完毕，产生脚本的时候根据规则识别出需要关联的动态内容，从而产生正确的脚本。
- 录制完毕回放一遍，把回放结果与录制结果进行自动对比，确定动态信息，进行自动关联。
- 录制两个一模一样的脚本，对比其中的差异来确定需要关联的动态信息，然后进行关联。

自动关联的功能是否完整可靠，关系到我们能否借助 Vugen 快速开发出符合要求的脚本，因此关联也是 Vugen 中非常重要的功能。

七、脚本的问题

Vugen 产生的最终结果是以源程序方式存在的脚本。为了编译该脚本，用户可以选用对应的编译器，这不是 Vugen 的功能。建议 Vugen 产生脚本的时候应该生成对应的 Makefile 和 build.xml，允许用户以流行的 make 和 ant 命令来编译 C 和 Java 的脚本。关于 make 和 ant，读者可以在互联网上查询相应的内容。

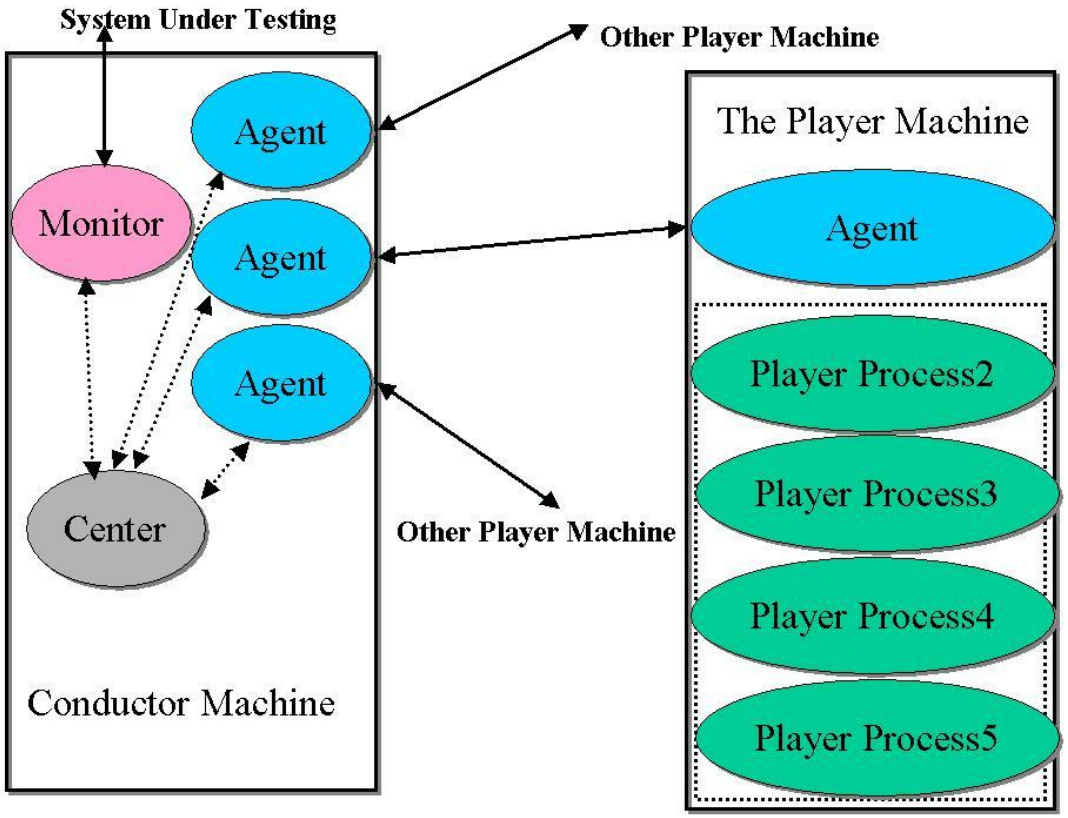
Vugen 自动产生的脚本应该支持两种语言，C / Java。很显然，Vugen 不可能产生一个脚本运行的全部的代码，它需要额外的函数库的支持。譬如，通过录制 Tuxedo 协议产生的脚本应该是以 Tuxedo-API 的形式出现的。为了能够编译运行脚本，必需把 Tuxedo 的函数库连接到脚本里面。目前动态库的技术应用非常广泛，因此为了运行 Tuxedo 脚本，必需在 Vugen 和 Player 机器上安装相应的 Tuxedo 客户端软件，因为它包含相应操作。其它网络协议也存在这个问题。对于 http 协议，已经有很多函数库。Vugen 产生的 http 脚本应该支持主流的函数库。这样带来的好处是我们不需要自己开发 http 函数库，可以

直接引用已经经过实践证明了的质量可靠的函数库。选择支持何种函数库，需要慎重选择，我们应该选择应用最广泛的函数库。例如：关于 http 函数库，可以采用 www.w3c.org 提供的 libwww，该函数库是开源的，质量可靠，远胜于我们自己开发。

八、Conductor 和 Player 部分

Conductor，我们称为“指挥家”，它是整个压力测试的核心。Player 是产生压力的负载产生器，它们以进程或者线程的方式运行由 Vugen 生成脚本。Player 如何运行脚本，由 Conductor 来决定。这好比一个交响乐队在演奏。Player 就是各种管弦乐演奏者， Conductor 是指挥者。

Conductor 和 Player 实际上是一套框架程序。具体执行什么功能，是由脚本来完成的。Conductor 和 Player 的体系结构如下图所示：



如上图所示，Conductor 上面有若干进程/线程。每种进程的作用如下：

- Center 进程是整个调度的核心进程，它负责联系和用户界面打交道的工作。
- Agent 进程负责和远端的 Player 机器中对应的 Agent 进程通讯。负责把编译好的脚本传送到 Player 机器上。在脚本运行

的时候，定期从 Player 机器上获取 Player 的运行状态，每个虚拟用户运行的日志。

- Monitor 进程负责对被测试系统的各个环节进行监控，并把监控的内容一方面写入 Conductor 机器的本地磁盘，另外一方面把监控的内容传送给 Center 进程，实时地显示在用户界面上。

Player 的进程有两种，一个是 Agent 进程，一个是 Player 进程。Agent 负责和 Conductor 机器通讯，它根据 Conductor 的指示，在本机器上派生出指定数目的 Player 进程，这些 Player 进程负责具体执行相应的脚本。Player 进程个数就是虚拟用户的个数。

Player 需要解决的一个问题是 IP 问题。为了防止黑客的攻击，某些后台的负载均衡设备一旦发现来自某一个 IP 的请求特别频繁时，就会拒绝为该 IP 提供服务。这样的功能造成的结果是 Player 无法把真正的压力加到后台系统中。解决方法就是在 Player 机器上伪装多个 IP 地址发送请求。这项技术称为 IP 欺骗(IP Spoofling)。Conductor 和 Player 必需实现该项功能。

九、Conductor 和 Player 的技术要点

关于 Conductor 和 Player 的技术要点有哪些，目前我还没有做深入的研究工作，但是我认为其技术要点主要涉及多进程/线程的编程，网络编程技术。可能这里面最大的难点是监控问题。当把被测系统的各个环节都监控起来，需要监控的参数会有成百上千个。如果采用集中式监控的方式，采集数据本身就对系统造成很大的影响，所以必需支持分布式监控方式。由于采集的数据是来自不同机器上的，由于各种的延迟，数据之间的时间同步将是一个重大的问题。

关于监控，还需要进一步的研究。

由于 Player 是没有界面的，是后台运行的程序，为了保持其可移植性，建议采用 Java 语言开发。

Player 和 Conductor 之间的网络协议不一定重新开发，可以使用成熟的 Http 1.1，方便在性能测试时调试 Player 和 Conductor 之间可能出现的通讯问题。

十、数据分析工具 Analysis

该工具是一个纯数学工具软件，目前市场上已经存在了大量负责数据处理的软件，如 Matlab 等。可以将压力产生的数据直接导入其

中进行处理。所以只要提供开放的数据接口就可以了，无需自己开发独立的性能数据分析软件。

即使 Analysis 需要开发，应该开发一些知识分析的功能。譬如，我们搜集了很多 Oracle 的数据信息，这些数据之间往往有固定的联系。如果将这些联系的知识融入到 Analysis 当中，将会更好。但是这有点类似人工智能的意味，比较难。

十一、 结束语

本文是对性能测试工具的一般性论述，讨论了性能测试工具的基本功能和可能出现的技术要点。由于性能测试工具涉及的内容太多，作者只是大致论述。其中涉及细节当中仍然会有很多技术要点没有论述。只是希望本文对希望了解性能测试工具的读者有一个入门的帮助。

一套功能全面的性能测试工具就象水管工经常携带的工具箱，里面充满各种工具，这些工具经过组合可以完成任何复杂的机械工作。完全从头开发这套工具箱，工程浩大，靠业余的编程爱好者是很难完成的。但是我们应该吸取 Unix“小而灵活”的哲学思想，在一个大的框架下面开发或者利用已经存在的开源工具软件制造出一个个灵活的部件。当把这些部件组合起来以后，就是一个功能完整、质量可靠的性能测试工具箱。

关于 SQAGetProperty 的使用

苏　扬

Arnold_Qian@MSN.com

【摘要】随着软件测试的发展，越来越多的人开始使用自动化测试工具帮助进行软件的测试，其中用的比较多的是 Rational Robot 和 Winrunner。由于 Rational robot 在测试 Web 方面比 Winrunner 有优势，所以许多人使用 robot 进行网站的测试。由于经常需要根据网站的一些信息来判断，决定脚本的运行情况，所以属性读取函数 SQAGetProperty 的使用很频繁。本文结合实际介绍了 SQAGetProperty 函数的用法。

【关键词】软件测试　自动化测试　Rational Robot　SQAGetProperty

一、序言

由于经常有人在论坛上询问有关如何读取网站上的信息，如何得到网页的返回值等等。所以想在这里全面介绍 SQAGetProperty 函数的用法，以及一些注意事项。

二、SQAGetProperty 函数介绍

SQAGetProperty 是用来抓取控件的属性的，脚本中可以根据所抓取的值来判断运行情况，然后执行不同的步骤。它的语法是 status% = SQAGetProperty(recMethod$, property$, value)。

recMethod$是如何读取你所要获得的控件的值的方法。在 recMethod$中，经常需要使用 Type＝来区分控件的类型，或者是使用符号来表示控件所在的层次关系。

property$是你所要读取的属性的名称，它是区分大小写的。

value 是个变量，是用来存放你读取的控件的属性值。

这条语句可以写成 SQAGetProperty “recMethod”, “property”, value

也可以写成 Result=SQAGetProperty(“recMethod”, “property”, value)

我是倾向于用后一种方法，因为方便调试。可以将断点设在该语

句上，调试的时候打开 Variable window 就可以看到 Result 的值了。

常用的 Result 值有：

Result=0，表示 SQAGetProperty 语句正确，能够成功读取属性的值。

Result=1002，表示 recMethod 的语法是错误的，需要改正你的语法。

Result=1003，表示你所要读取的控件没有找到，说明 recMethod 部分的语法还是对的。这种情况经常出现在抓取含有网页层次关系的控件中。如果网页层次关系没有表示好，就会出现找不到控件的错误。

Result=1005，表示你要读取的属性没有找到。可能是你想抓取的值并不是控件的属性，也可能是在区分控件前的“\”丢失了。

SQAGetProperty 函数的应用范围

用 SQAGetProperty 函数所读取的控件的值，是要可以用 Object Properties 读取出来的，否则是不能通过工具直接抓取控件的值的。

对于那些直接写在<body></body>里的文字，如 HTMLDocument 是无法用该函数抓取的。

比如 <html>

<head></head>

<body>Hello</body>

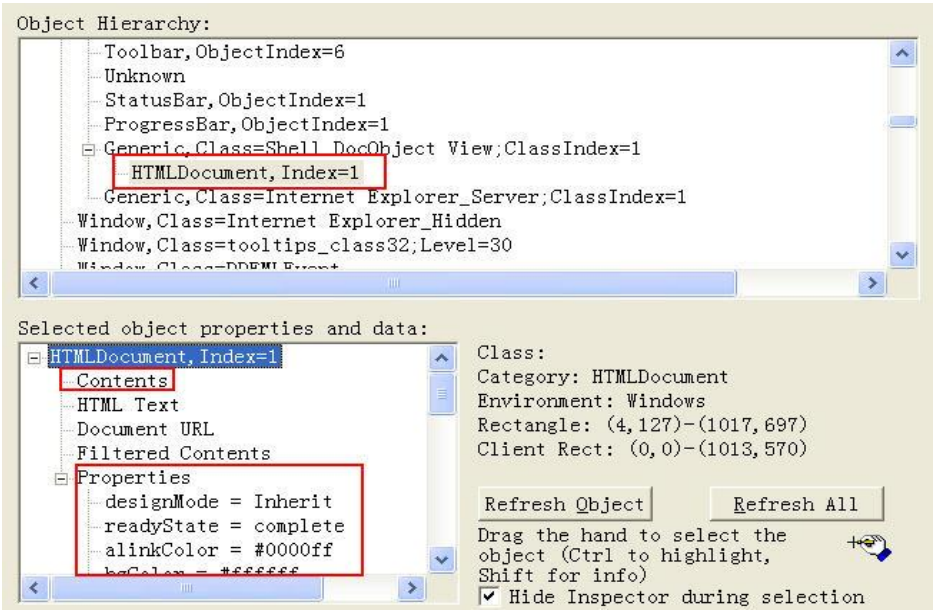
</html>

这个 Hello 就没办法用 SQAGetProperty 抓取出来，它也没有办法用 Object Properties 抓出来，因为它并不包含在 HTMLDocument 的属性值中。

如何判断一个内容是否可以用 SQAGetProperty 抓取出来呢？建议使用 robot 本身的辅助工具——insqector 进行分析。

打开 inspector，选取你希望取出的内容进行分析，可以得到控件的列表。

抓取上面所说的 Hello，所得如图：



由图可见，该 Web 页面只有一个 HTMLDocument，它的 Index 为 1。而 Hello 两个字只是在 Contents 里的。可是 HTMLDocument 只有 Properties 里的属性才能用 Object Properties 抓取出来，而只有那里面的属性才能用 SQAGetProperty 读取出来。也就是说如果 robot 可以取到该对象属性，那么说明 robot 提供的函数 sqagetproperty 也可以取到该属性的。这是判断 SQAGetProperty 是否能够成功的依据之一。

三、举例说明 SQAGetProperty 函数的用法

由于 SQAGetProperty 函数主要取决于 recMethod$部分的书写结构，下面我将以是否含有网页的层次关系，举例具体的说明 recMethod$部分的书写方法。

四、对于不含网页层次关系的控件属性的读取

对于这类控件，recMethod$部分很简单，可以仿照帮助里的例子。

比如：

Result = SQAGetProperty("Type=CheckBox;Text=Match case", "State", CheckState)

就是将类型是 CheckBox 的，内容是 Match case 的控件，将他的 State 属性读取出来，放置在 CheckState 这个变量里。

我们拿一个实际的例子来说：

51Testing 论坛登陆成功后，会出现当前用户的名称。我们可以

读取 HTMLTable 的信息来验证是否出现了该用户名。

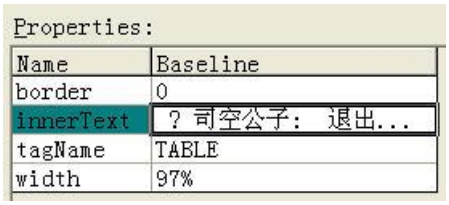
首先用 Object Properties 抓取所需要的控件。抓取后显示如下：

Window SetTestContext, "Caption=51Testing 软件测试论坛---软件测试,软件质量工程师的家园 - Microsoft Internet Explorer", ""

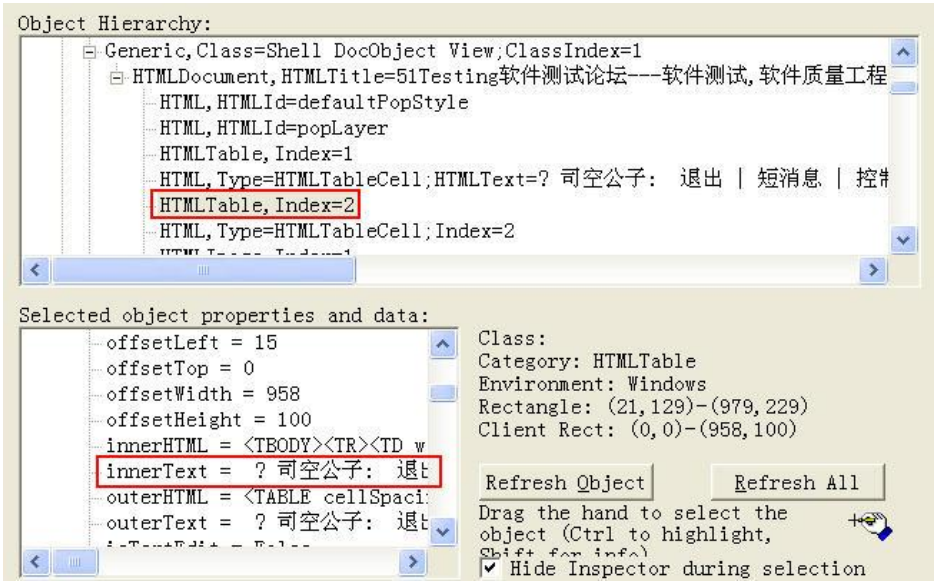
Result = HTMLTableVP (CompareProperties, "Index=2", "VP=Object Properties")

Window ResetTestContext, "", ""

可以查看验证点，得知所要取的值在 innerText 里，如图：



用 inspector 工具同样可以知道所要获取的值的属性，并且可以确切的了解所要获取的控件是否含有层次结构。如图：



由图可见，所要获取的控件是 HTMLTable 类型，它没有层次关系，并且 innerText 属性是在 Properties 中的，是能够用 SQAGetProperty 函数读取出来的。

编写脚本，可得

Sub Main

Dim Result As Integer

Dim varStr as String

Window SetTestContext, "Caption=51Testing 软件测试论坛---软件测试,软件质量工程师的家园 - Microsoft Internet Explorer",

""

' Result = HTMLTableVP (CompareProperties, "Index=2", "VP=Object Properties")

' Window ResetTestContext, "", ""

Result=SQAGetProperty("Type=HTMLTable;Index=2","innerText",varStr)

End Sub

此时读出的 varStr 值为：“? 司空公子: 退出 | 短消息 | 控制面板 | 系统设置 | 搜索 | 统计 | 帮助”

这样只需要再通过其它一些函数比如 Left$、Right$之类的对该字符串进行加工，就可以得到所需的用户名。

一点说明：不能将 Window SetTestContext 给注释掉，否则会导致 robot 无法定位到所需的网页上，也就无法读取到控件。返回错误为 1003，找不到控件。

五、对于含有网页层次关系的控件属性的读取

大部分网页没有复杂的层次关系，但对于很多软件的 Web 客户端，尤其是使用了 struct 之类架构的 Web 客户端来说，网页层次关系就比较复杂了。此时使用 SQAGetProperty 函数来抓取控件的属性时，就必须要注意 recMethod$部分的写法是否清楚的表示了网页的层次关系。

很多人之所以用这个函数无法取出数值，都是因为 recMethod$中的层次关系没有表示清楚。我们可以通过 inspector 工具查看一个控件的具体的层次关系。

我们可以借用 PCL 兄在《請問 Robot 如何快速使用座標定位》中所构建的框架结构来说明这个问题。

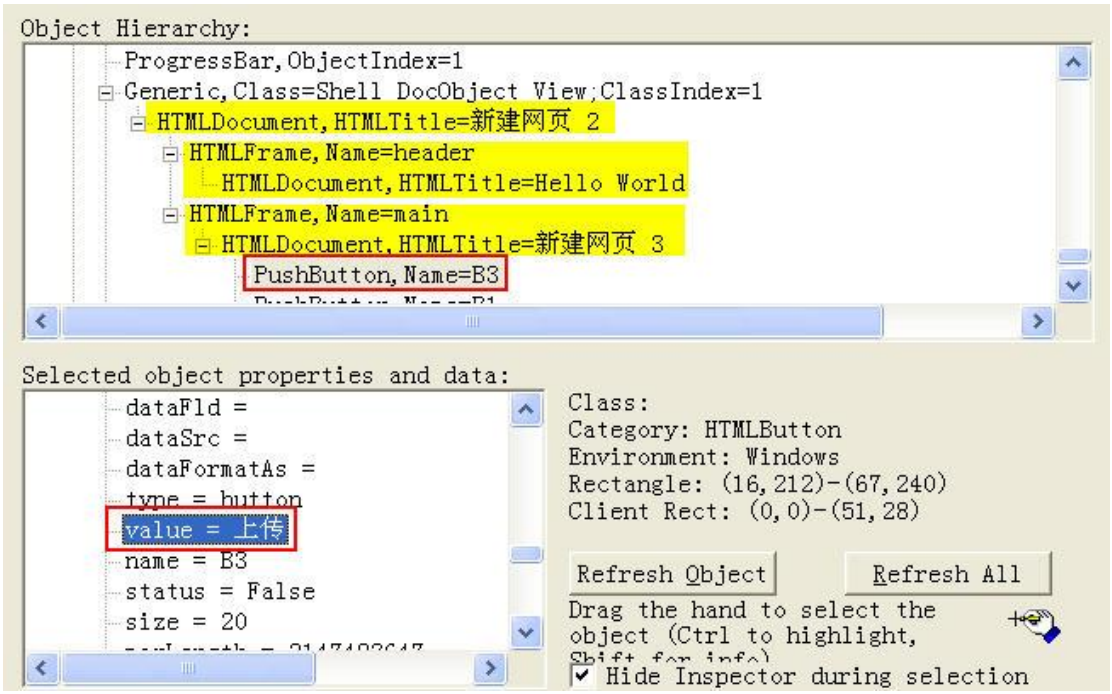
首先搭建框架代码。

Frame.htm:

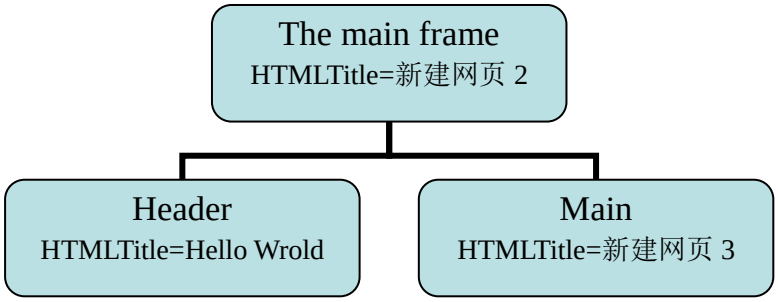
```
<html>
<head>
<title>新建网页 2</title>
</head>
<frameset rows="64,*">
  <frame name="header" scrolling="no" noresize target="main"
```

```
src="hello_world.htm">
        <frame name="main" src="new_page_3.htm">
        <noframes>
        <body>
        <p>此网页使用了框架，但您的浏览器不支持框架。</p>
        </body>
        </noframes>
    </frameset>
    </html>

    Hello_world.htm:
    <html>
    <head>
    <title>Hello World</title>
    <base target="main">
    </head>
    <body>
    <p>Hello World</p>
    </body>
    </html>

    new_page_3.htm:
    <html>
    <head>
    <title>新建网页 3</title>
    <SCRIPT FOR="B3" EVENT="onClick"
LANGUAGE="VBScript">
        MsgBox "上传文件！"
        </SCRIPT>
    </head>
    <body>
    <form method="POST" action="--WEBBOT-SELF--">
        <input type="button" value="上传" name="B3">
        <input type="submit" value="提交" name="B1">
        <input type="reset" value="重置" name="B2"></p>
```

加入我们想获取“上传”这个按钮上的字符，可以使用 inspector 工具查看，如图：



可以看出，这个框架包含两层。



表示层次关系时用“;\;”来划分。

比如想读取“上传”这个按钮的内容，可以首先用 Object Properties 抓取这个 button。

得结果为：

Window SetTestContext, "Caption=新建网页 2 - Microsoft Internet Explorer", ""

Browser SetFrame,"Type=HTMLFrame;HTMLId=main",""

Browser NewPage,"HTMLTitle=新建网页 3",""

Result = PushButtonVP (CompareProperties, "Type=PushButton;Name=B3", "VP=Object Properties")

Window ResetTestContext, "", ""

然后可以使用 SQAGetProperty 函数编写脚本为：

Result = SQAGetProperty("Caption=新建网页 2 - Microsoft Internet Explorer;\;Type=HTMLFrame;HTMLId=main;HTMLTitle=新建网页 3;\;Type=PushButton;Name=B3", "Value", StateString)

分析 recMethod$部分。“Caption=……”表示最外面的一层框架，让 robot 首先定位到“新建网页 2”这个页面。紧接着“;\;”表示层次的划分。“Type=HTMLFrame;HTMLId=main;HTMLTitle=新建网页 3”使 robot 定位到 main 这个框架里。最后的“Type=PushButton;Name=B3”定位到所需抓取属性的 button 上。

如果在调试过程中发现始终有 1003 的错误，可以在 SQAGetProperty 语句前加

Window SetTestContext, "Caption=新建网页 2 - Microsoft Internet Explorer", ""

Window ResetTestContext, "", ""

进行定位。

总结

本文着重介绍了 SQAGetProperty 函数在网页控件属性方面的抓取，以及如何通过 Object Properties 和 inspector 工具在区分网页的层次结构，使得能够比较方便的写出正确的 SQAGetProperty 语句。

本文关于 SQAGetProperty 函数的使用介绍到此结束，希望大家能有个满意的答案。要想掌握好这个经常使用的函数，必须要多做实验，掌握在各种情况下的语法形式。

如果还有什么疑问，或者是希望对 robot 的其它部分进行讨论的，可以联系我。我的 MSN 是 Arnold_Qian@MSN.Com

参考文献：

1. Rational Robot User Guide

2. 论坛帖子《請問 Robot 如何快速使用座標定位》、《robot 如何取页面输入域的文本内容》、《如何让 combox 自动选取它的子项》

针对办公自动化系统软件的测试分析方法

吴丽莎

【摘要】目前 OA 系统软件在软件项目中占有一定的比重，本文主要针对 OA 系统软件需求，给出相应的软件测试分析的基本思路。

【关键字】办公自动化系统、软件需求

一、办公自动化系统的简介

办公自动化即 Office Automation，简称 OA。

目前流行的办公自动化系统多采用多层体系结构，其应用服务架构位于中间层之上，客户端通过常用的 IE 浏览器界面访问系统，具有接口统一、访问简单、易升级、易扩充的特点。

就以上特点来说，办公自动化系统的测试可以使用 B/S 结构的测试策略来组织。

下面我们就从软件工程过程的几个阶段—需求、设计、编码，分阶段地来进行分析如何针对办公自动化系统组织测试分析。

二、针对 OA 需求、设计的特点组织测试分析

办公自动化系统擅长处理类似公告、公文等流转类型的行政办公类应用需求、设计及相对独立的个人相关资料、名片、记事等个人事务类的需求、设计。

办公自动化系统软件的权限管理是其不同于其他应用软件的另外一个特点。系统需要为使用人员提供设置不同的权限和访问许可的功能，管理员可以通过调整各功能模块的访问权限，设置一般用户某些功能可以用，某些功能不允许用；并为员工创建、注销帐号及访问权限。提高了企业系统的资料的安全度，阻止非授权人的非法进入系统。

1. 针对流转型的行政办公需求、设计

流转型的行政办公需求、设计通常包括有：拟稿、审核、签收、会签、拟办、签批、电子签名、交办、退稿、备查、提交归档、打印、销毁等业务处理功能。在行政办公的业务流程处理中还牵涉到复杂的用户权限和访问许可的功能。

针对这种情况，在进行测试需求分析和设计时，最好使用现成的

公司体制来进行分析。这样做的好处有两个：

1) 沟通方便。

现成的公司体制中的角色和人员比每个测试人员自己单独构造虚拟的用户权限和访问许可要容易理解和容易沟通的多。

对于测试执行过程中发现缺陷时，描述的缺陷让开发人员和测试人员沟通起来更加方便。

2) 测试数据准备容易，且不易产生歧异。

由于 OA 系统使用的对象是整个公司所有员工或者是某部门员工，如果我们使用现成的公司体制，我们只需要统一准备一套测试数据就可以满足所有测试对象的要求。

测试人员在沟通时，不会由于构造的数据不同，而引起不必要的歧异，人为的增加测试组内部沟通的障碍。

2. 针对独立型的个人事务需求和设计

独立性的个人事务通常包括有：维护并查看个人和公共的活动日程安排，并能自动提醒所有个人的待办事项，允许用户可以查询各种信息。但个人事务通常只允许用户本人维护和查看个人的事务，不允许其他用户维护和查看。目前有的 OA 系统中甚至包括管理员在内的超级用户也不能维护和查看私人的个人事务处理情况。

针对上面的特殊情况，在进行测试需求分析和设计时，首先要考虑统一给不同用户打上特殊标记。接下来在准备测试数据时，应避免不同用户具有相同的个人信息和相关资料的情况产生，以免在测试执行过程中测试人员陷入混乱状态，连测试人员自己都搞不清楚到底使用的是哪一个用户的信息。

三、针对 OA 编码的功能特点组织测试分析

常见的 OA 系统功能模块主要有：行政办公、个人事务、综合信息、基础服务四个部分。

1. 行政办公

行政办公通常包括收文管理、发文管理、档案管理和会议管理等模块。有的 OA 系统还包括有接待管理、办公资源管理模块。

这四个模块是典型的流转型模块，它们都有流程定义、登记（或拟稿）、办理、拟稿、审核、签收、会签、拟办、签批、电子签名、交办、退稿、备查、提交归档、打印、销毁、归档、查询、流程跟踪、

查看意见、重定位等操作过程。

以收文管理为例，主要对公文进行登记和处理，包括内部公文和外部公文。在登记收文过程中提供了多种种方式，比如文件引入、电子公文调入、扫描和直接输入，并将登记后的收文送领导批示或阅读（批示的流程完全可以根据用户的需要自己定义，也可以使用系统管理员已经定义好的公文批示流程），处理结束后将文件进行归档。管理人员可以对收文处理全过程进行监督、催办、重定位，也可以随时进行文件流程跟踪及查看其所有领导的批示意见、批示时间。

针对这些情况，在进行测试分析和设计时，首先按照上面提到的根据现成的公司体制进行分析和设计的测试数据，然后将各个领导是否兼职的情况区分开来。通常建议准备这样两套数据：

1) 领导不兼职

领导不兼职的情况，相对较简单，即每个领导只负责一个批示。

在执行测试过程中，还需要重点注意批示的并行和串行的情况。

2) 领导兼职

领导兼职的情况，即每个领导可能负责不同过程中多个批示，是流转型模块测试的一个难点，需要特别注意。

跟上面的情况一样，同时也要考虑批示的并行和串行的情况。在测试执行过程中，其组合方式是否能够全面覆盖，与测试人员的经验、对模块的需求和设计熟悉程度、测试数据准备是否充分以及测试人员是否考虑周到全面等因素息息相关。

2. 个人事务

个人事务通常包括：待办工作、日程安排、个人资料、个人名片、个人记事本、外出声明等模块。有的 OA 系统还包括个人邮件、及时消息模块。

个人事务以其独立性，完成个人日常的办公工作，例如批阅各部门上报的各种公文，评阅同事交流的各种文件内容，回复或发送电子邮件，起草各类报告，查看个人的活动日程、外出等安排，系统能自动提醒待办事项。

以个人名片为例，用户可将名片登记并进行管理查询和打印，同时可根据需要将部分名片共享，供他人使用。每个人只能看到自己的名片集及共享的名片集，通过所有由个人收集的名片以及整个单位的名片总集，可很快找出所需要联系的名片主人，并方便地通知他们参加会议或发送邮件等等。

在进行测试分析、设计和执行中需要特别考虑：

1) 新建或修改的名片时对于输入重复的名片是否给予提示警告；
2) 新建或修改的名片时个人维护的私有名片是否能被其他人看到或使用；
3) 个人删除私有名片时是否影响到其他用户的名片；
4) 共享的名片是否可以被其他人正确查看和使用；
5) 单位的名片集修改后，是否正确影响个人的单位名片集；
6) 给需要联系的名片主人联系时，是否可以正确联系上，其联系内容是否显示正确；

3. 综合信息

综合信息通常包括：建议管理、电子论坛、网上调查、电子贺卡、信息采集等模块。

以信息采集为例，信息采集可以通过各种渠道，从所有可利用的信息源收集办公需要的信息，从各种媒体采集各种相关信息后作为原始信息记录在案，经过筛选整理后编辑成各种主题的信息刊物。同时信息刊物也支持套红头转入行政办公的公文模块中。可以方便地查询、检索信息刊物及其所有原始信息内容。并对信息采用和阅读情况、次数进行统计。

在进行测试分析、设计和执行时要重点考虑：

1) 从信息来源收集信息时，是否能正确完好的保存其原始信息的内容和格式；
2) 整理后的信息是否能正确完好的保存其原始信息的内容和格式；
3) 整理后的信息是否能正确转入公文流程中；

4. 基础服务

基础服务包括：人员注册、部门设置、数据维护等模块。

以数据维护为例：系统为系统的管理员提供了多项数据维护的服务。可以对一些常用的数据进行设置，包括用户登录名/用户密码组合方式、用户登录名/用户密码长度、主题词、常用意见、自动编号、存储大小、存储时间和公文格式，也可以对行政办公中所要使用的各个流转模块的流程进行预定义。

在进行测试分析、设计和执行时要特别考虑：

1) 用户登录名/用户密码组合方式设置是否正确；
2) 用户登录名/用户密码长度设置是否正确、有效；
3) 存储大小设置是否正确、有效；对于超出设定的存储大小系统是否能正确提示；
4) 预定义的行政办公中各个流转模块是否能被正确应用；

四、小结

OA 系统的某些业务与其他知识管理系统相类似，但由于其鲜明的特点，目前已经自成体系。

本文介绍的测试分析主要与 OA 特有的业务处理方法紧密联系，作为测试人员介入 OA 项目时如何有重点的进行测试分析。

与其他 B/S 结构的系统所要进行的界面测试、边界测试、非法校验、字段限制等方法一样，在实际执行测试过程之前都需要一一进行分析，在此就不赘述了。

作者简介：

吴丽莎，1999 年毕业，2001 年进入软件测试行业，在西安多家大公司担任软件项目测试主管。至今积累了丰富的软件测试实践经验，熟悉 ISO9000 和 CMM 体系。

性能测试原理及性能测试实例分析

柳　胜

【摘要】在大型软件系统投入生产之前进行性能测试已经成为趋势，本文结合一个性能测试案例对性能测试的过程和原理进行了介绍。

【关键字】性能测试 并发测试 负载测试

一、软件测试中的性能测试

软件测试是保证软件质量的重要手段，也是软件过程中一个必不可少的环节。而性能测试则隶属于软件测试中的系统级测试，它对软件在集成系统中运行的性能行为进行测试，旨在及早确定和消除软件中与构架有关的性能瓶颈。

1. 性能测试的含义

目前对性能测试没有明确的定义，一般地，它主要是针对系统的性能指标制定性能测试方案，执行测试用例，得出测试结果来验证系统的性能指标是否满足既定值。性能指标里可能包括系统各个方面的能力，如系统并发处理能力，批量业务处理能力等。

2. 性能测试的分解

在性能测试的执行中，可以根据具体的性能指标，分解为几种测试，根据其关系，可以在不同的时间和空间内执行。这些子测试通常包括以下几种：

并发测试：验证系统的并发处理能力。一般是和服务器端建立大量的并发连接，通过客户端的响应时间和服务器端的性能监测情况来判断系统是否达到了既定的并发能力指标。

负载测试：验证系统的负载工作能力。系统配置不变的条件下，在一定时间内，服务器端在高负载情况下的性能行为表现。这里的负载可以是用户数，交易数，事务数等。

配置测试：核实在操作条件保持不变的情况下，系统在使用不同配置时其性能行为的可接受性。

健壮性测试：核实被测系统的性能行为在异常或极端条件之下的可接受性。这里的异常或极端条件指的是资源过少，用户数过多，突发故障等。

随着软件系统的规模日益庞大，结构日趋复杂，对软件系统的性能测试已经成为必须和趋势。尤其大型的分布式软件系统更要在正式运行前进行性能测试，因为这样的系统在投入生产之后，往往要接受大批量的业务量，这对应用程序本身，操作系统，中心数据库服务器，中间件服务器，网络设备的承受力都是一个严峻的考验。在其中任意一个环节出现的问题都可能给用户带来巨大的商业损失。 预见软件系统的并发承受能力以避免商业风险，这是在软件测试阶段就应该解决的。例如中国人民银行的现代化支付系统和上海外汇交易中心的本币交易系统都在投入生产之前进行了多轮的第三方性能测试，起到了很好的作用。

下面我就介绍一个性能测试案例。

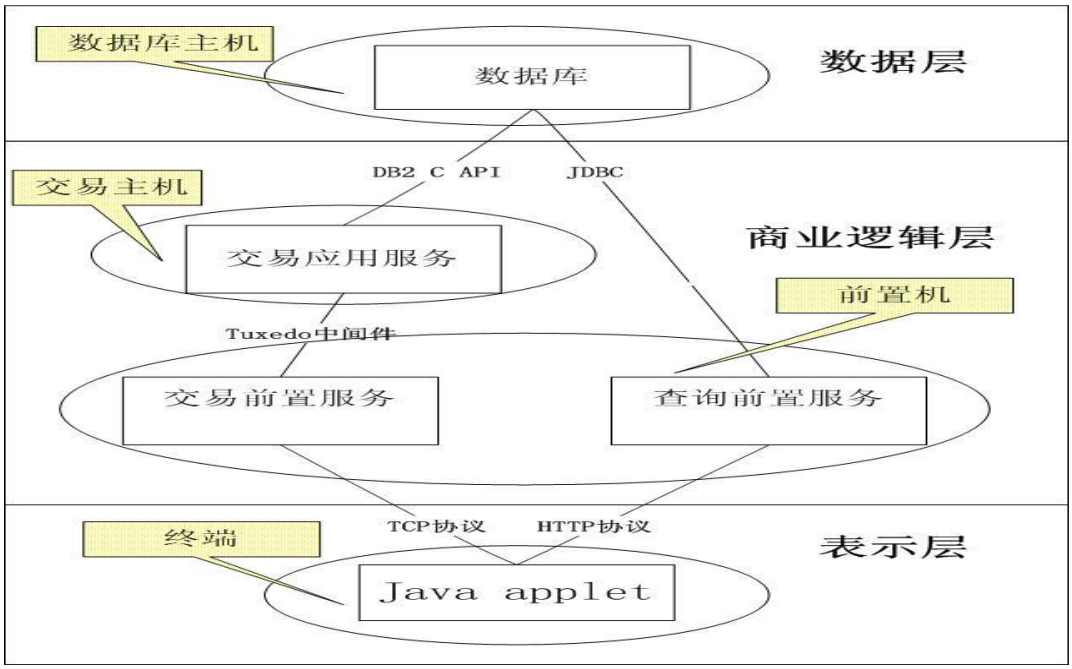
二、一个性能测试实例

1. 被测系统

1) 被测系统介绍

本系统应我国金融信息化发展设计，采用当今比较先进和流行的技术，是运行在城域网上的大型分布式应用系统。

本系统遵循 J2EE 规范，采用 B/S 体系结构进行设计和开发。业务主要分为交易业务和查询业务，查询业务采用 J2EE 规范，交易业务以 J2EE 体系架构为基础，进行进一步的处理，采用了 TCP 的四层结构。系统体系结构图如下：



图表 1 被测系统体系结构设计图

A. 表示层：

运行在终端上。运行java applet程序，提供协议控制和用户界面，与系统最终用户实现直接交互，通过TCP/HTTP与前置系统通讯。向前置系统发送请求报文，并接收前置系统返回的回应报文。

B. 商业逻辑层：

作为中间层实现核心业务逻辑服务。

- 交易应用服务：运行在交易主机上。在tuxedo中间件上运行业务处理程序，按交易规则处理前置机发来的交易指令，通过tuxedo jolt与前置机连接，通过DB2 C API与数据库连接。
- 交易前置服务和查询前置服务：运行在前置机上。交易前置服务运行服务程序接收终端请求报文并通过tuxedo jolt客户端将其转发给交易主机，再通过轮询和同步反馈接收交易主机返回的报文，将其转发给业务终端；查询前置服务运行在weblogic应用服务器上并调用Jreport组件，通过JDBC完成对查询流指令的发送并接受数据库返回的结果给业务终端。

C. 数据层：

运行在数据库主机上。负责整个系统中数据信息的存储、访问及其优化。运行DB2数据库服务程序。通过DB2 C API与交易主机通讯，JDBC与查询前置服务通讯。

数据库主机和交易主机运行在交易中心城市，前置机运行在各个分中心城市，终端是各个城市参加交易的单位，整个系统覆盖城域网。

2) 被测系统的性能要求和性能指标

金融系统是业务处理十分频繁、数据交换吞吐量很大的系统，业务处理的速度直接关系到公司的经济效益和客户对公司的评价。在客观条件下，整个广域网系统必须在大业务量的情况下同时保持快速的实时响应能力，以保证整个业务系统的通畅运行。用户对此提出如下性能指标：

指标种类		用户需求
登陆		系统能够处理750用户/分钟，至少支持上百用户并发 登陆响应时间不超过30秒
业务	交易	系统能够处理300笔/分钟 响应时间不超过10秒
	查询	系统能够处理400笔/分钟 响应时间不超过10秒

表格 1 用户要求性能指标表

下面我们会根据此系统和给定的性能指标来进行性能测试：

2. 对被测系统进行性能测试

性能测试的目的是最大程度地模拟真实业务场景，来验证系统的性能指标，并发现可能存在的性能瓶颈。

1) 对被测系统进行系统分析

我们可以看到本系统大体上由终端、前置机、交易主机、数据库主机节点组成。

在整个业务流程中，业务终端→前置机→交易主机→数据库主机形成了一个压力流串，每个节点在压力下能够正常工作是整个系统正常运转的基础。也就是说，如果其中任意一个节点在业务压力下发生了拥塞、处理不力等不正常情况，那整个系统都无法正常运转。

我们来看一下业务流程。

首先，从终端到前置机，终端产生业务报文发送至前置机，前置机上运行查询前置服务和交易前置服务，查询前置服务向下通过HTTP 协议以 WEB 服务形式和终端连接，向上通过 JDBC 直接与数据库系统相连。交易前置服务向下通过基于 TCP 协议的 socket 连接和终端通讯，向上通过 tuxedo jolt 客户端和交易应用服务连接。交易应用服务进行业务逻辑计算，并操作数据库系统。

由以上分析，我们可以整理出整个系统的两条压力流程线来，之所以我们把其分为两条流程线，是因为交易前置服务和查询前置服务的工作原理完全不同，下与终端的连接，上与交易主机的连接也完全是独立的两个通路。

终端→交易前置机→交易主机→数据库系统

终端→查询前置机→数据库系统

下面我们先独立分析两条流程线，之后我们将再次综合分析，以考虑二者之间的相互影响作用。

第一条路线上主要运行的是登陆指令和交易指令信息。

当系统运作时，多个交易终端与交易前置服务建立socket连接，完成登陆，之后发送交易指令，造成对交易前置服务的压力。交易前置服务通过运行服务程序接收到交易指令，并检验其合法性，然后通过交易中间件tuxedo的客户端把业务的压力传递给交易主机进行处理。交易主机进行必要的金融计算和业务逻辑运行，得出反馈结果，生成消息，一方面顺原路返回到各个终端上去，一方面记录入数据库。

在本条流程线上的加压主要考验交易前置服务程序的socket多连接建立能力，tuxedo交易中间件的即时响应能力，交易主机的计算能

力，以及DB2数据库的DML语句加锁机制。

第二条路线上主要运行的是查询指令信息。

查询指令产生时，通过http协议访问weblogic上的web服务器和应用服务器上的相应组件，以JDBC接口访问后台的DB2数据库，并把数据库返回的结果发送至终端界面。

在本条流程线上的加压主要验证weblogic处理能力，数据库中索引是否创建合理。

两条流程线相对独立，但又是互相依赖的。由于是对同一个数据库系统进行读操作和写操作，查询流程的结果依赖于交易流程数据的产生，交易流程的产生的数据又通过查询流程得到验证。在进行压力测试时，两者的协同会对数据库形成压力的冲击。

鉴于以上分析，结合用户性能指标，我们决定把本次性能测试分解为如下几个子测试来进行。

A. 并发登陆测试：750个终端一分钟内并发登陆系统，并且响应时间在30秒之内。

B. 业务负载测试：

此下又有三个子测试。

- 交易流程测试：多个终端发起交易请求，逐渐加压，以达到300笔/秒的压力为限。
- 查询流程测试：多个终端进行查询，逐渐加压，以达到400笔/秒的压力为限。查询成功与否以所请求的web页面完全展现为标准。（查询响应能力其实和数据库中的数据量有关系，后来和用户进一步确认，基础数据为30万条）。
- 综合测试：

在上面两种测试都通过的情况下，进行综合测试。

2) 性能测试的执行过程，性能测试依照下面的步骤来进行：

第一步：测试脚本的开发

本次压力测试采用MI公司的loadrunner工具，脚本编辑和编译工作在VU Generator(脚本作坊)中进行。

理想的脚本是对现实世界的业务行为进行了完全无误的模拟，这其实是不可能的。我们的目标是使模拟的误差在我们认可的范围之内，并能有方法加以控制。

针对并发登陆测试和交易流程测试，由于两者运行机理相同，都

是终端调用socket client，和交易前置的socket server建立连接，将请求消息发送至交易前置机。我们考虑采用将此部分java socket程序编入测试脚本程序，生成登陆和交易业务脚本，通过loadrunner来执行。这样做的好处是绕过终端IE界面复杂的处理逻辑，直接施压在前置机上（这种方式同时也带来了偏差，在执行测试场景时通过其它方法得到了一定的弥补）。

脚本除了要实现与前置机的socket连接，业务发送等功能，还要建立用户信息数据池，设置检测点、异常退出点，为脚本执行后的结果统计和分析提供正确的依据。

交易业务脚本内容略。部分如下：

```
public class Actions
 {
                    /*    登陆变量初始化*/
        ProtocolManager protocol;    //ProtocolManager 为实现 socket 连接的类
         ServiceName service; //ServiceName 对服务端的信息进行了封装，包括 IP 地址和端口号。
              LoginMessage login; //LoginMessage 为登陆时需要向服务器发送的消息，待服务器确认并返回回应消息时，登陆成功。
          protocol = new ProtocolManager(); //创建 ProtocolManager 类的 protocol 对象
              service = ServiceName.getInstance();//获得 ServiceName 的实例
              login=new LoginMessage();//创建 LoginMessage 类的 login 对象
         service.setIP("200.31.10.18");//设置服务端的 IP 地址
         service.setPort(17777);//设置服务端的端口号
                 /*设置登陆消息*/
              login.serUserName(lr.eval.string("{loginName}"));//从数据池里读出用户名，设置在 login 成员变量里

              login.setPasswd("1234");//数据库中添加的用户密码都为 1234
                /*发送登陆消息*/
              protocol.login(login);//发送登陆消息
```

```
lr_start_transaction("trade");//交易开始点
TradeMessage trademessage;//生成交易消息
  /*设置交易消息*/
    ……
    ……
  /*发送交易消息*/
    ……
    ……
 if(sendfail)
  lr_end_transaction("trade", LR_FAIL);//如果发送交易消息失败，交易结束，返回。
  /*循环回收主机返回的处理信息*/
    ……
    ……
 if(recievefail)
    lr_end_transaction("trade", LR_FAIL);//如果不能接收到主机处理回应消息，交易结束，返回。
 if(recievesuccess)
    lr_end_transaction("trade", LR_PASS);//如果接收到主机成功处理的回应消息，交易结束，返回。
    ……..
}
```

在上面的例子中，我们主要对每笔交易进行了transaction化。在交易开始时设置开始检测点，交易结束时设置结束检测点，并给loadrunner报出交易状态。实际的脚本中在回收交易响应消息时还进行了拆包，在应用层上对交易状态进行识别，并非例子中只在socket层加以判断。

针对查询流程测试，由于loadrunner工具支持基于http的web访问录制功能，我们将考虑采用以录制脚本为主，手工编写脚本为辅的方法，生成查询业务脚本，通过loadrunner来执行。由于查询脚本基本由录制生成http请求和应答，不同的压力测试工具录制会有差别，这里就不再写出查询脚本样例。

第二步：根据用户性能指标创立测试场景

在本次性能测试中，用户提出的性能指标不够细致和确切，通过

对用户调查和实际业务分析，我们把性能指标的实现方式进行了明确的定位。

A. 并发登陆测试场景

并发登陆 750 用户/分钟，登陆响应时间在 30 秒之内。仔细考虑一下，这里的并发登陆 750 用户/分钟指的是系统能够在 1 分钟内接受 750 个用户的登陆请求，而处理的效果如何则在交易终端体现，即登陆响应时间。基于这样的理解，我们把用户性能指标转化为如下的测试场景：

从第一秒钟开始，用 loadrunner 每秒钟登陆 13 个用户，并保持 socket 连接，直到 1 分钟结束，从终端向系统一共发送 750 个左右的用户登陆请求，系统在一分钟内建立了 750 个连接。在终端观察并统计登陆响应时间。如果系统不能响应持续增加的登陆请求或平均登陆响应时间大于 30 秒，并发登陆测试场景都不能算通过。

为了帮助用户更加深入了解系统的能力，我们对系统的瞬时并发能力进行测试，即测试系统所能承受的最大的瞬时并发用户登陆连接请求个数。这个场景通过 loadrunner 在登陆前设置同步点来实现，这个结果将结合上一个结果一同反映系统的登陆处理能力。

B. 交易流程测试和查询流程测试：

在这里我们只对系统的业务负载能力做测试（并发处理能力在登陆测试中已经得到考证）。测试场景如下：

在 loadrunner 中，建立 goal-orented 的测试场景，以 400 笔/秒为目标，将调度权交给 loadrunner 来试图达到这个指标。

C. 综合测试：

交易流程测试和查询流程测试同时进行。

以上的测试场景要求均可在 loadrunner 中的 Controller 进行设置完成。

测试场景的创建之后，我们的测试任务更加具体化和清晰化。

第三步：运行测试场景，同步监测被测系统性能行为

在 loadrunner 中的 controller 中开启 unix 系统资源计数器，weblogic 计数器，DB2 计数器，检测系统资源消耗情况，并最终和测试结果数据合并，成为分析图表。

测试结果可在测试执行完毕后，通过 loadrunner 工具中的 Analysis（分析器）获得。

A. 并发登陆测试：

依照设计好的测试场景，用 loadrunner 工具在一分钟内渐增向系统发送登陆请求。分别进行三次，结果如下：

测试序列	登陆成功的用户数	平均响应时间
第一次	485个用户	25.6秒/笔
第二次	497个用户	27.3秒/笔
第三次	482个用户	26.2秒/笔

表格 2 登陆测试结果数据表

注：这里的登陆成功用户指的是系统接受了登陆请求，并建立了连接。平均响应时间在登陆脚本里设置检测点，由 loadrunner 工具自动获得。

考察系统的瞬时并发处理能力：在完成上一步测试的前提下，逐步增加瞬时并发登陆用户数，直到系统极限。

测试执行结果如下：

瞬时并发用户数	计划并发数	1	2	10	100	200
	实际并发数	1	2	10	100	122
用户登陆响应时间	最短（s）	2.0秒	3.1秒	7.2秒	21.2秒	24.2秒
	平均（s）	2.0秒	3.3秒	12.2秒	24.3秒	28.1秒
	最长（s）	2.0秒	3.5秒	16.5秒	31.2秒	33.3秒

表格 3 瞬时并发登陆测试结果数据表

B. 负载测试：

✓ 交易流程测试：

测试结果如下：

对于通过网络接口发送的批量业务请求，均在性能指标所指定的时间范围内得到请求成功的反馈消息，说明主机已经处理成功。

在通过网络接口发送业务请求的同时，开启 IE，通过实际终端界面进行登陆和交易，系统响应时间延长，界面显示和刷新明显变慢，到业务量高峰时期，界面已经不能显示任何信息，处于不可工作的状

态。

需求说明的是系统正常工作时，每个界面终端不仅应该能够展示己方的交易信息，还要展示其他交易单位的交易信息和系统信息。因此当交易量大的时候，界面需要展示的信息量是巨大的，这本身对终端界面是一个性能考验。

✓ 查询流程测试：

本流程测试在交易流程测试之后进行,以利用其生成的数据。

测试结果基本满足性能指标。

✓ 综合测试

由于交易流程测试的未通过，本测试已经不能执行。

第四步：测试结果分析及性能评价

A. 并发测试结果分析

根据上述的并发测试响应时间表，我们可以得出以下的结论：

被测系统在一分钟内并不能接受 750 个用户的登陆请求，其可接受的登陆请求用户数大概为 490 个左右。在这样的条件下，登陆响应时间在用户要求范围之内。

被测系统的瞬时并发处理能力约为 122 个用户。

B. 交易流程测试结果分析及性能评价

根据交易流程测试结果可知，通过脚本程序进行业务行为，发送业务请求消息到回收主机处理回应消息， 这段时间系统是顺畅的，反应也是迅速的，但是在终端界面却不能即时展现消息。这说明信息的回馈通路在终端界面出现了性能瓶颈。当界面需要在短时间内展示大量交易信息时，已经不能承受负荷。这与终端采用 java applet 技术有关。

C. 查询流程测试结果分析

查询流程基本符合性能指标。

需要说明的是，实际中，以上每个场景的测试都执行了多次，中间件参数进行了多次的调优。从以上测试的结果分析也可以看出，我们的性能测试瓶颈不是出现在中间件产品上，而是在自身开发的程序上。

三、总结

由以上的实例过程我们可以看出性能测试基本由以下几个步骤

进行

1. 系统分析：

将系统的性能指标转化为性能测试的具体目标。通常在这一步骤里，要分析被测系统结构，结合性能指标，制定具体的性能测试实施方案。这要求测试人员对被测系统结构和实施业务的全面掌握。

2. 建立虚拟用户脚本：

将业务流程转化为测试脚本，通常指的是虚拟用户脚本或虚拟用户。虚拟用户通过驱动一个真正的客户程序来模拟真实用户。在这一步骤里，要将各类被测业务流程从头至尾进行确认和记录，弄清这些交易过程可以帮助分析到每步操作的细节和时间，并能精确地转化为脚本。 此过程类似制造一个能够模仿人的行为和动作的机器人过程。这个步骤非常重要，在这里将现实世界中的单个用户行为比较精确地转化为计算机程序语言。如果对现实世界的行为模仿失真，不能反映真实世界，性能测试的有效性和必要性也就失去了意义。

3. 根据用户性能指标创建测试场景：

根据真实业务场景，将单个用户的行为进行复制和控制，转化为多个用户的行为。在这个步骤里，对脚本的执行制定规则和约束关系。具体涉及到交易量，并发时序等参数的设置。这好比是指挥脚本运行的司令部。这个步骤十分关键，往往需要结合用户性能指标进行细致地分析。

4. 运行测试场景，同步监测应用性能：

在性能测试运行中，实时监测能让测试人员在测试过程中的任何时刻都可以了解应用程序的性能优劣。系统的每一部件都需要监测：客户端，网络，web 服务器，应用服务器，数据库和所有服务器硬件。实时监测可以在测试执行中及早发现性能瓶颈。

5. 性能测试的结果分析和性能评价

结合测试结果数据，分析出系统性能行为表现的规律，并准确定位系统的性能瓶颈所在。在这个步骤里，可以利用数学手段对大批量数据进行计算和统计，使结果更加具有客观性。在性能测试中，需要注意的是，能够执行的性能测试方案并不一定是成功的，成败的关键在于其是否精确地对真实世界进行了模拟。

在整个性能测试过程中，自动化测试工具的选择只能影响性能测试执行的复杂程度，简便一些或繁杂一些；但人的分析和思考却会直接导致性能测试的成败。所以本篇着重于对性能测试思路的整理。测试工具的介绍可以参看有关压力测试工具的资料。

注 1：在本次性能测试案例中，还涉及到健壮性测试和可恢复性测试，限于篇幅，只介绍了并发测试和负载测试。

注 2：loadrunner 脚本样例并非实际运行脚本，只是为了表示其流程。

参考文献：

Roger S. Pressman：软件工程实践者的研究方法 黄柏素 梅宏译 机械工业出版社

软件测试和第三方测试服务的需求调查

殷德 上海浦东软件园评测中心

【摘要】本文主要对软件测试的和第三方测试的调查结果进行系统分析，得出一些软件测试外包市场相关的结论，提供给业内人士参考。

【关键字】软件测试、外包、第三方测试

一、软件测试现状

软件测试包括测试技术、测试方法和测试管理。软件测试是软件质量的核心。

对信息化依赖程度不断加深的社会，必然会对软件质量提出全方位的要求---安全、稳定可靠、方便灵活。软件测试正是控制软件产品质量的重要手段，控制软件产品质量的重要手段就是通过权威机构的软件测试。国外软件厂商极为重视软件测试。为打造 windows2000，微软用了 250 多个项目经理、1700 多个开发人员，而测试人员则用了 3200 人！几乎是开发人员的两倍。而且，每修改一个错误，都要花费大量时间确保没有新错误产生。

目前，我国软件业的质量保证体系还很不完善。相比之下，在国外许多国家的软件公司，软件测试工作已经逐渐演变成一门独立的科学，囊括了配置方案，测试机制，跨平台策略和产品性能，稳定性等独立区域的知识模块。

长期以来，我国软件企业产品开发时，测试成本却常常是最容易被压缩，甚至被完全“砍”掉。这导致我国软件产品质量低下，无法创出自己品牌，走向世界。在国际上，开发成本中的 30-50%用于软件测试。而国内有的开发成本远远达不到这个百分比。上海市计算机软件评测重点实验室的评测报告显示，在 2002 到 2003 年评测的 560 多项软件产品中，一半以上的软件文档质量不规范。与此同时，只是有少数的软件企业设立了专门的测试部门。因此，在当前不断加深于对外合作的环境下，除了优秀的软件开发团队，具备良好的软件测试环境也是软件市场与国际接轨的必不可少的重要部分。

目前国内的软件测试一般有下列几种形式：

1) 软件公司内部的功能性测试，目的是检查设计的功能能否完

成；在软件开发管理规范的软件公司，软件测试比较全面规范，质量工作可以说完成的比较好。在软件开发管理比较薄弱的公司，测试是由开发人员或者抽调同公司别的部门的人员进行的。或者干脆不做测试。

2) 用户测试，大量的用户一起寻找使用中遇到的错误。发现的问题多集中在用户使用方面，及可用性方面问题。用户测试，因为用户没有系统的计算机知识，对产品的测试结果也只是片面的，不能全面对软件产品进行评价。

3) 第三方测试，就是运用专业软件测试管理机制，专业的测试人员运用一定的测试方法、工具对软件的质量进行全面检测。其形式更多的为：软件本地化企业组织测试人员到大型软件公司的软件开发现场进行测试。这是大多数软件本地化企业不愿意接受却又实际采用的模式，主要是因为软件开发商保证新项目信息保密安全，便于监控软件测试的进度和质量。

4) 据介绍，国际上软件开发人员与测试人员的比例大都在 1：1，软件测试收入占软件总产值的 20%，而国内软件产业尚未形成这种状态。截至 2002 年底，软件企业中通过 ISO9000 认证的企业仅占总数的 7.5%。不过令人高兴的是，国内大中型软件企业开始认识到软件质量的重要性。很多企业已经配备质量保证体系，以及企业内部的测试队伍。

目前，国内已拥有的第三方测试机构很少。但在软件业较发达国家，绝大多数的软件产品的认定，都需要第三方测试的介入，软件测试行业产值几乎占了软件行业总产值的 1/4。与之相比，国内的软件测试行业实在微乎其微。空白同时意味着机遇，潜力也许就在其中。

二、市场调查数据

那么在目前的情况下，国内软件企业对软件测试的重视程度和对软件第三方测试认知程度又是什么样的呢？

笔者在最近的几个月时间里，通过网络对目前的软件测试和第三方测试服务的现状做了相关的调查。通过反馈回来的数据我们可以看到国内目前的现状。

1. 调查内容

在本次调查中，调查的问题包括：

1) 公司规模：a、10 人以内；b、10~50 人；c、50~100 人；d、

100 以上。

2) 开发队伍人数：a、10 人以内；b、10~50 人；c、50~100 人；d、100 以上。
3) 是否有测试人员：a、是；b、否。
4) 测试队伍人数：a、0；b、10 人以内；c、10~50 人；d、50~100 人。
5) 公司对采用测试过程后，感觉是：a、很好，软件产品质量有很大提高；b、一般，没什么太大变化；c、很糟糕，还不如不测试；d、从没有测试过，不了解。
6) 是否听说过测试外包服务：a、是；b、否。
7) 贵公司是否需要测试外包服务：a、是；b、否。
8) 如果需要测试外包服务，您希望的方式是：a、服务方全部负责测试全过程；b、协助本公司完成部分测试工作既可；c、其它（请说明）。
9) 是否有完善的测试管理机制：a、是；b、否。

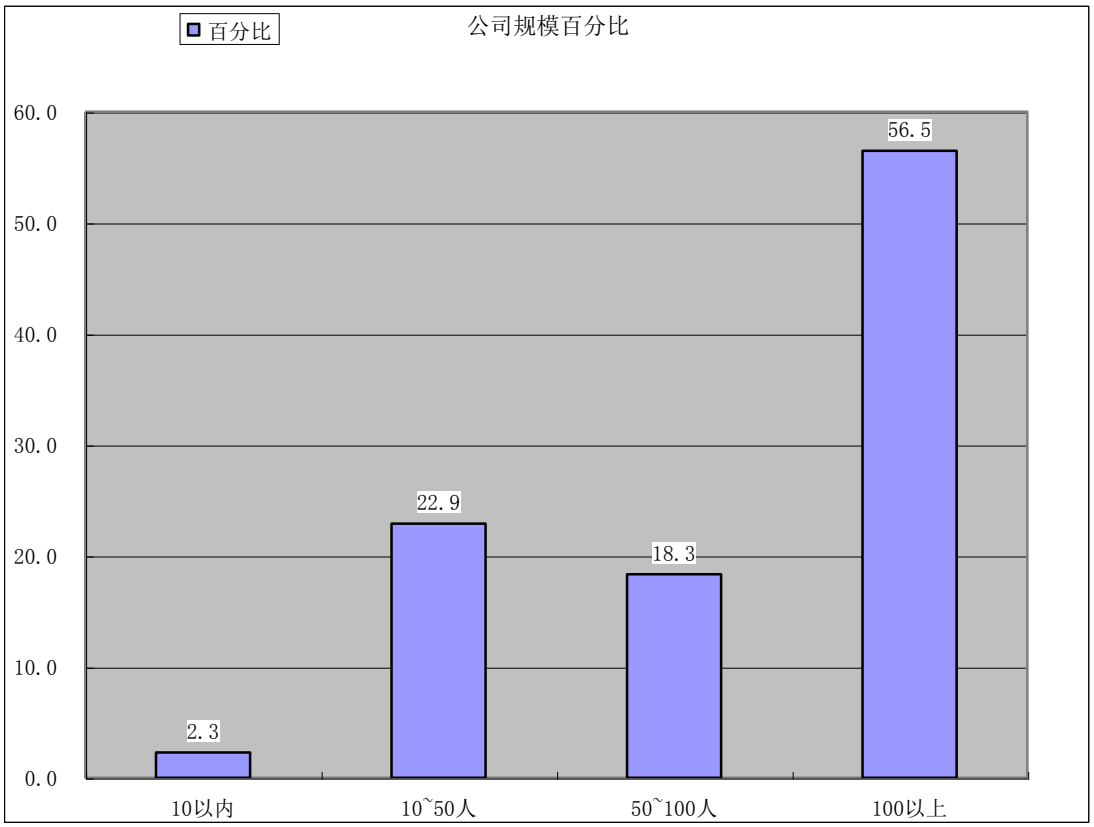
2. 调查目的

本次市场调查的主要目的：

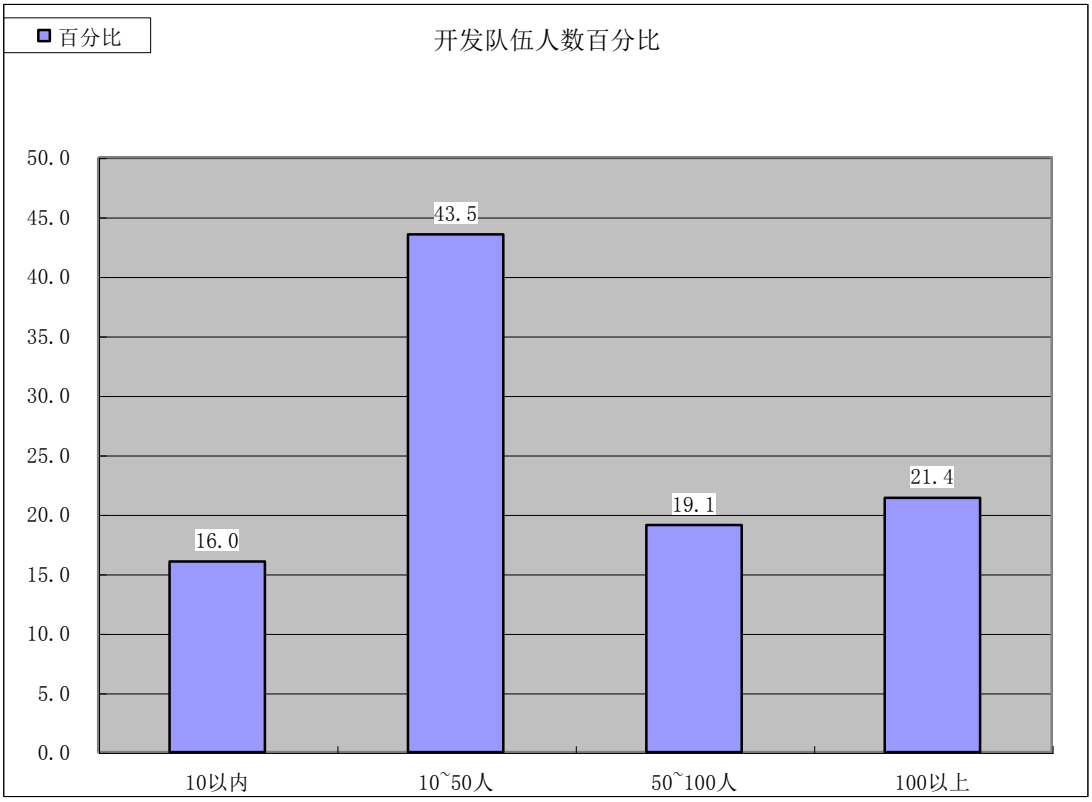
1) 国内不同规模软件企业对软件测试的重视程度；
2) 测试队伍和完善测试管理体系的普及率；
3) 国内不同规模软件企业对软件测试外包服务方面信息的了解程度，和企业是否愿意接受软件测试外包服务的情况；

3. 调查数据图表

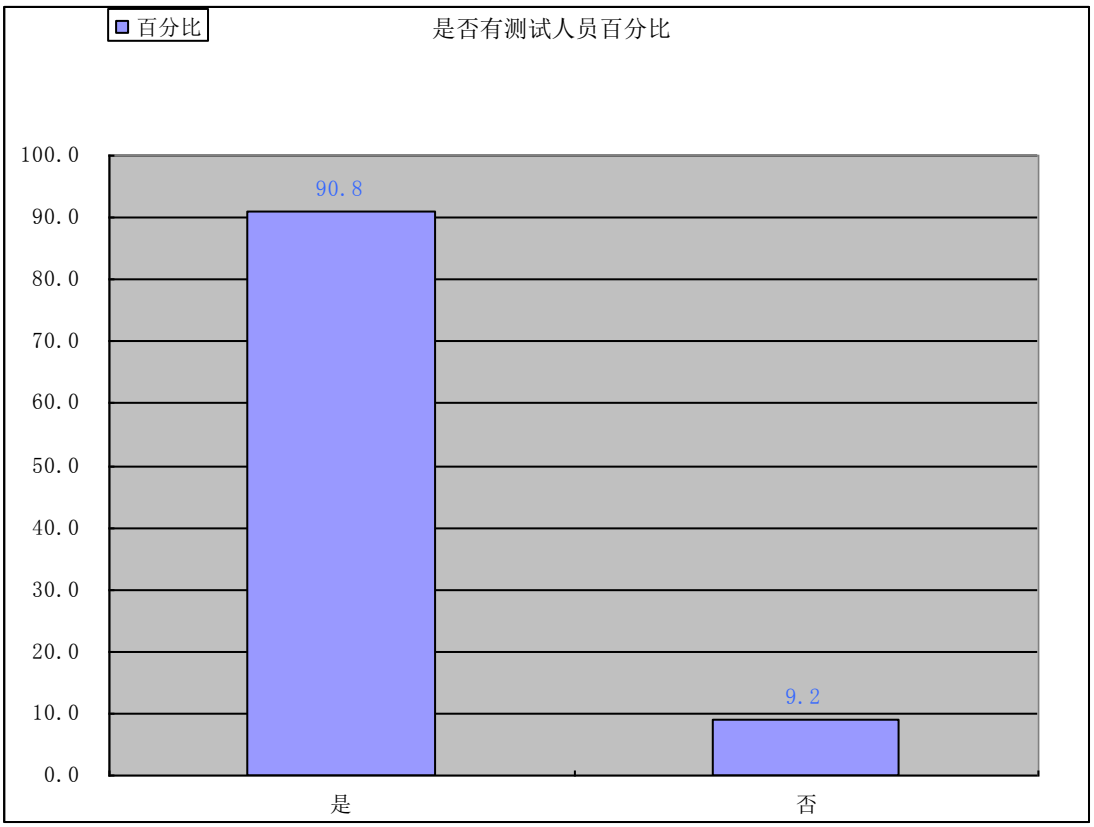
1) 公司规模



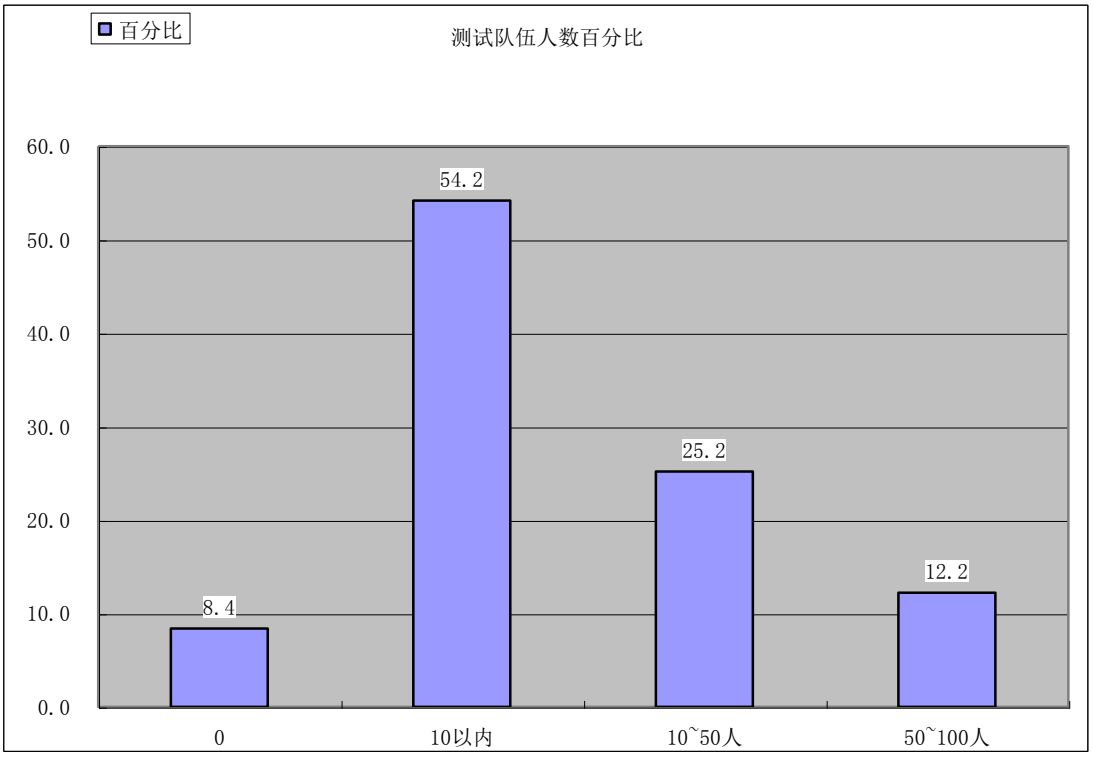
2) 开发队伍现状



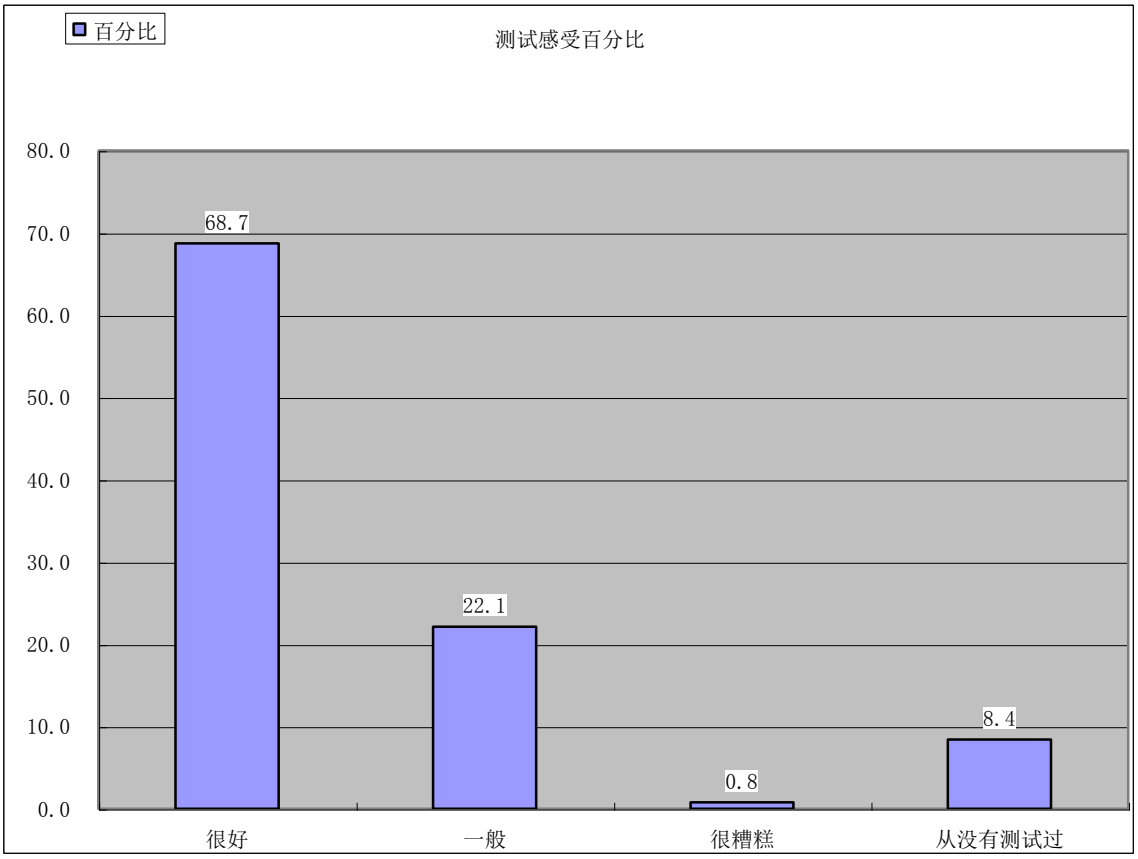
3) 公司是否有测试人员


百分比
是否有测试人员百分比
100.0
90.0
80.0
70.0
60.0
50.0
40.0
30.0
20.0
10.0
0.0
90.8
9.2
是
否

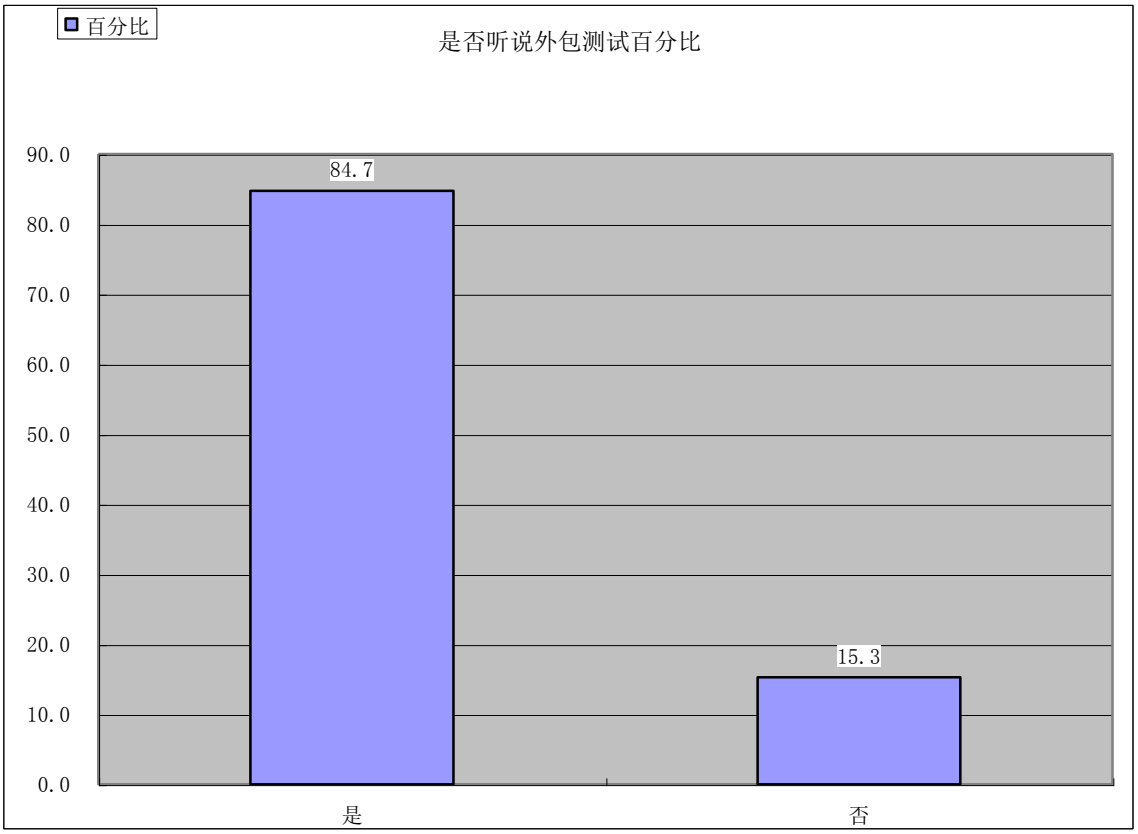
4) 测试队伍现状


百分比
测试队伍人数百分比
60.0
50.0
40.0
30.0
20.0
10.0
0.0
8.4
54.2
25.2
12.2
0
10以内
10~50人
50~100人

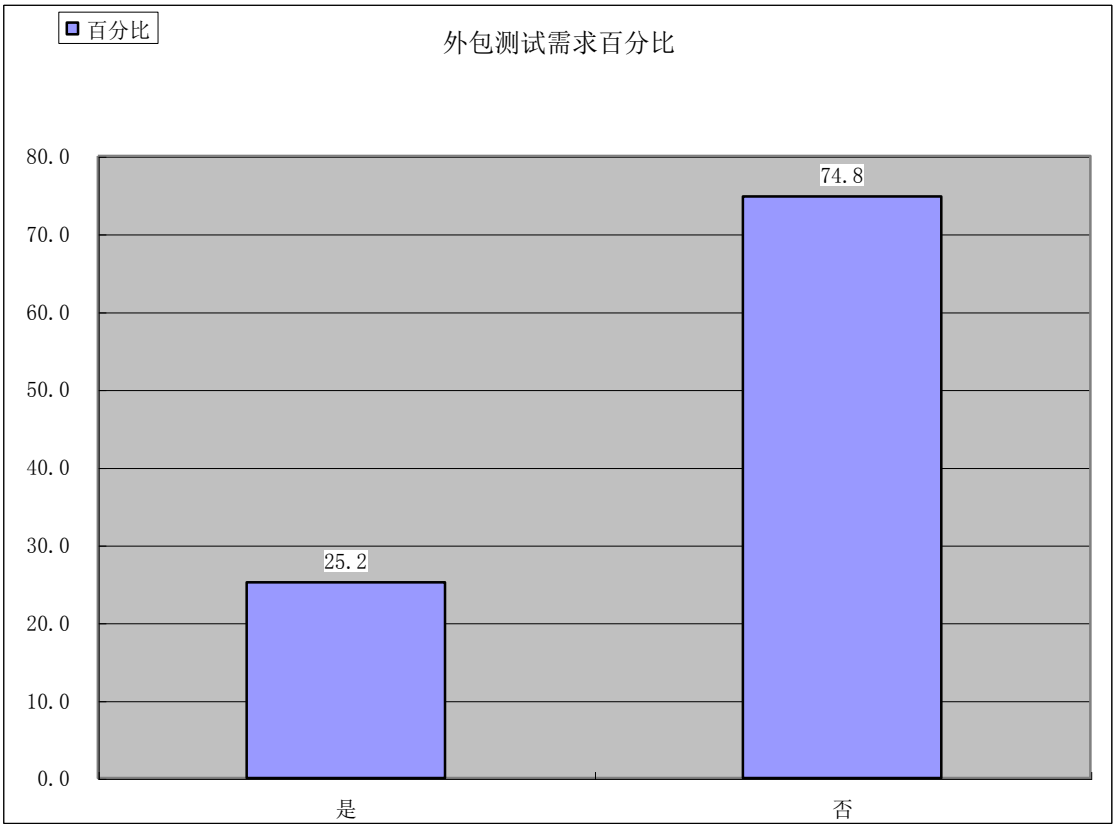
5) 公司采用测试后对测试的感受



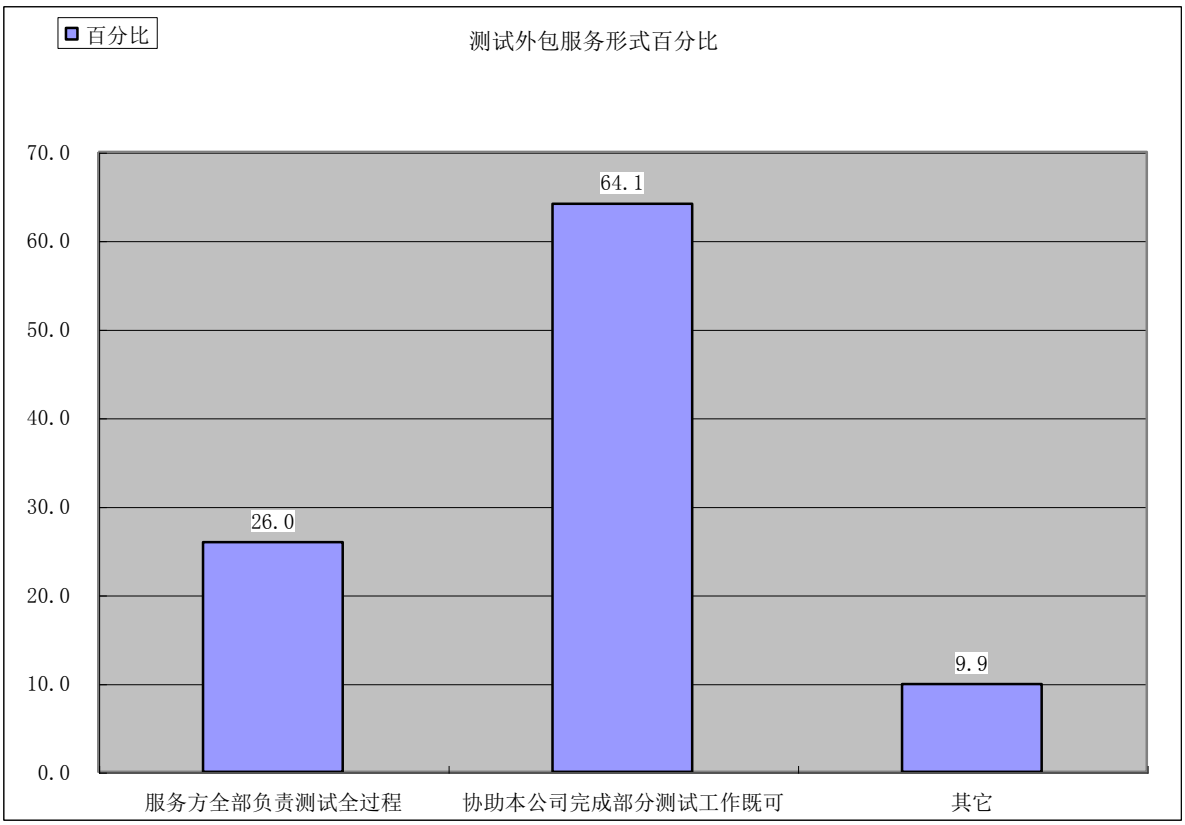
6) 是否听说过外包测试服务



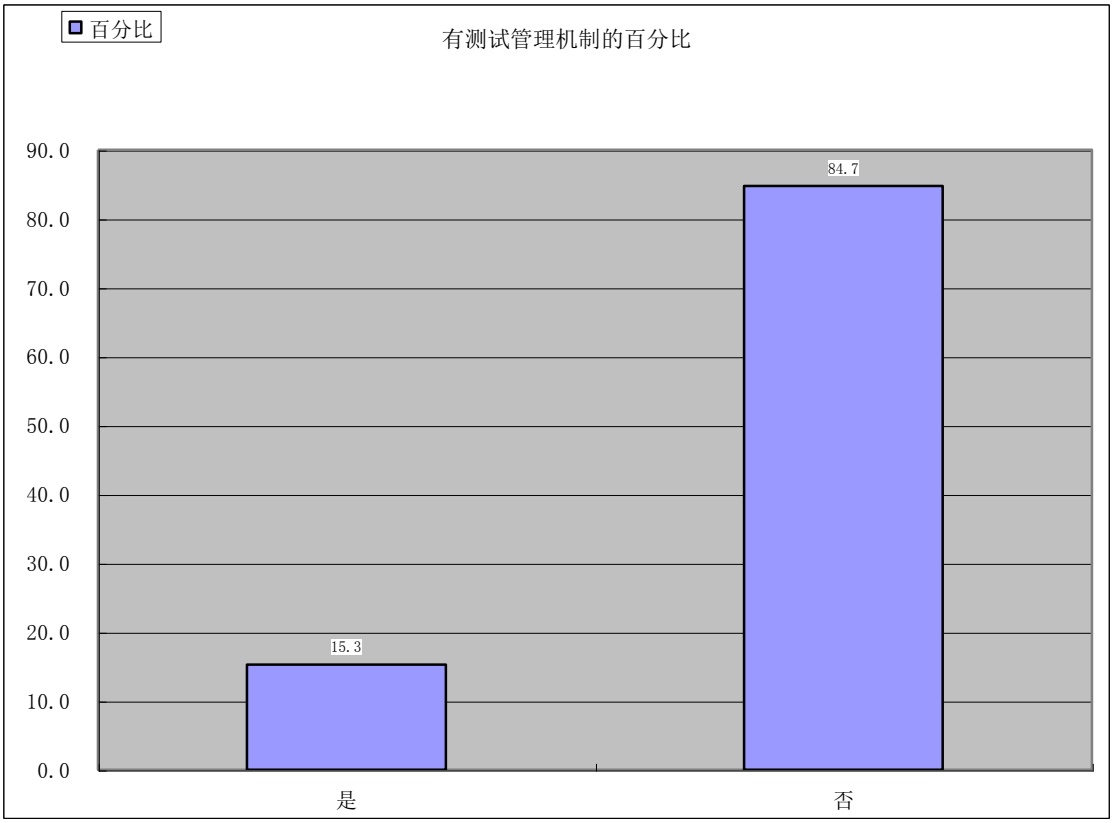
7) 外包服务需求



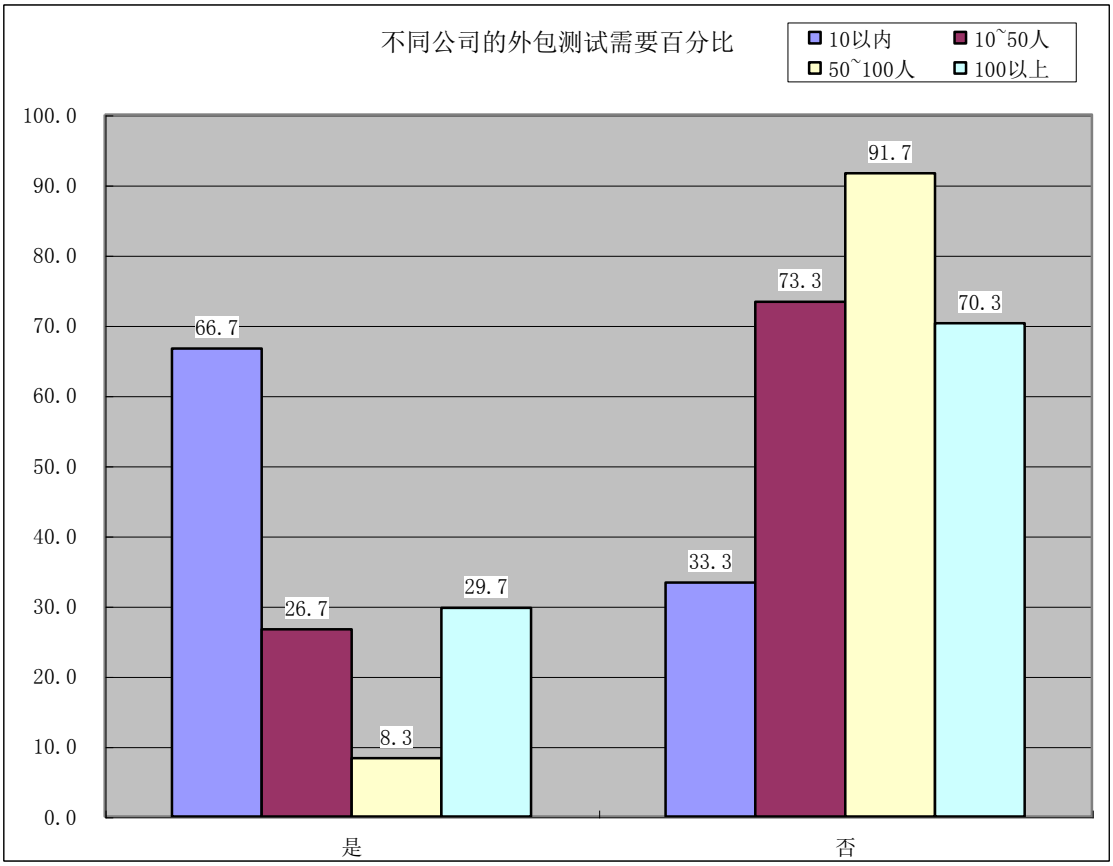
8) 外包服务需求形式



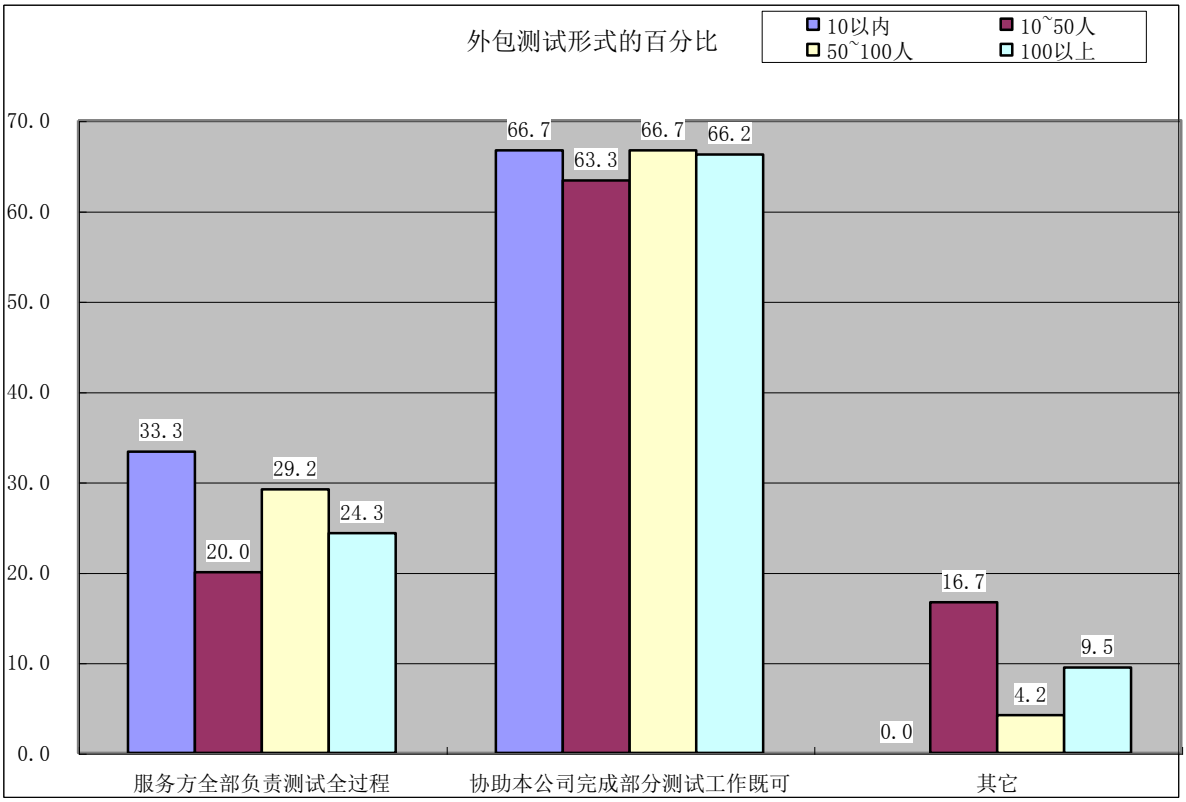
9) 是否有完善的管理机制



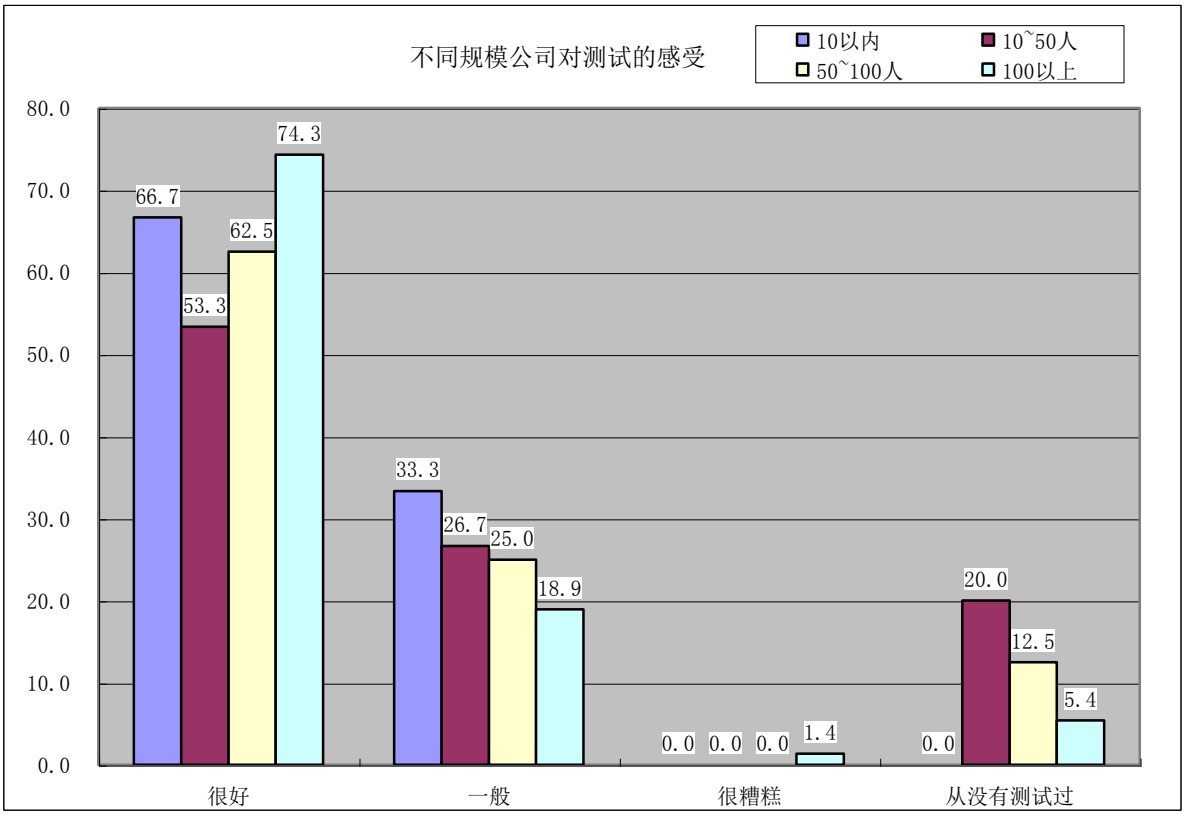
10) 不同规模公司的外包测试需求



11) 不同规模公司的外包测试形式



12) 不同规模公司对测试的感受



在本次调查中，来自全国各地的 130 多家软件企业参与了调查。

其中员工人数在 100 人以上的软件企业在这次调查中数量最多，占57%左右。其次是员工人数在 150 人以下的小型软件企业，占 25%左右。

通过网上的调查数据表明，目前国人的软件企业有近 91%左右已经配备了测试队伍，更多的企业通过软件测试来提高自身的软件产品质量。总体上，认为通过测试后软件质量得到很好提高的占 69%左右，认为一般的占 22%左右。在不同规模的软件企业对软件在测试后质量提高的认同比率都是非常高的，认为软件测试后对软件质量提高作用甚微的比率很小不足 1%。但我们还看到。在调查的所有企业中仍有近 8%左右企业根本没有对软件产品进行过测试。在所有的调查企业中，即便是在软件开发过程中通过软件测试方式来提高软件产品质量，但对于测试过程的控制还是重视不够。在测试管理方面具有完善的管理机制的仅占 15%左右。这个比例还是比较低。

外包测试服务的调查数据表明，在国内，外包测试服务的概念已经广泛的流传，也被许多企业熟识，有近 85%左右的调查企业了解外包测试。但是，虽然很多企业了解外包测试的形式，但愿意采用外包测试服务的企业并不多，仅占 25%左右。在愿意采用外包测试服务的公司中，从公司规模上看，10 人以下的软件企业愿意采用外包测试服务的需求最高。其次 100 人以上的软件企业比例稍大一些，约占30%左右。

在所有的调查企业中，假设采用外包测试服务的前提下，采用“协助本公司完成部分测试工作既可”这种形式的最多，占 64%左右。而同意采用“服务方全部负责测试全过程”仅占 26%左右。对于不同规模的软件企业普遍认同采用“协助本公司完成部分测试工作既可”这种形式，调查数据显示的百分比都超过 60%。该形式成为软件企业不愿意接受却又实际采用的模式，出现这种现象的原因是因为软件开发商保证项目信息保密安全，便于监控软件测试的进度和质量。

通过调查数据显示，国内软件企业已经逐渐重视软件测试在开发中的重要地位。但是因为处于发展阶段，在质量控制方面并没有找到很好的途径，有的还处于摸索阶段。很多企业没有很好的相应管理模式。在质量保证的实施方面，还处于“自力更生”阶段。对采用外包服务的模式，还没有被广泛的接受。

三、现状分析

1. 软件质量保证

在软件开发过程中，测试应占有很重要的地位。但为什么在实际的软件开发过程中测试的成本被压缩呢？根据国内目前的现状，分析原因有以下几点：

1) 国内软件用户（企业）对计算机使用率普遍低，即使在计算机设备比较普及的企业中，员工对计算机的了解大多数处于初级阶段，对软件测试的认识更少。
2) 计算机使用的范围受限，必然限制软件的推广使用，用户对于软件质量的认同标准不高，普遍认为能够使用即可，出了些问题也可以接受。正是由于客户的这种心理，使得软件开发企业敢于压缩测试成本。
3) 软件企业规模小，软件开发过程不完整。重开发，轻测试。有的软件公司根本没有测试队伍。
4) 软件企业在压缩测试成本的同时，对开发相关的支持文档也不重视。这为企业的发展带来很多不利的影响。
5) 软件企业对软件质量要求的认同存在差距，同时也缺乏有效的改善措施。有些软件企业，即使有测试队伍，但缺乏有效的测试管理，使得企业内部的测试工作效率低下。
6) 某些企业因为受到资源的限制，虽然想提高软件质量，但由于缺乏资金、人员等方面的资源，没有能力去做质量管理方面的工作。
7) 软件质量保证工作需要相关的人力资源、硬件资源、管理体系、软件测试工具等。

虽然，目前很多软件企业都配备了专门的质量保证部门或软件测试队伍，但软件测试的重要性还没有得到普遍的认同。但随着软件企业和软件用户群的质量意识不断提高，用户对软件测试服务的需求增大。提供软件测试服务还是有比较大的前景和市场。

2. 测试服务

目前，在国际上软件业较发达的国家，绝大多数的软件产品的认定，都需要第三方测试的介入，软件测试行业产值几乎占了软件行业总产值的 1/4。而国内在测试服务方面，软件测试服务的还处于起步和摸索阶段。专业的第三方测试机构还非常少。通过借助第三方测试完成软件开发的企业更少。

第三方测试机构都希望通过拓展市场，来扩大自己的生存空间。虽然，国内的测试服务市场还不够成熟，但在未来测试服务市场肯定

会有很大的发展空间。在目前还存在以下的问题：

1) 软件测试服务的对象不清晰。
2) 如何提高客户对软件测试服务的需求。
3) 如何提高测试机构的业务和技术水平？
4) 软件测试服务应该包括那些具体的内容？
5) 第三方测试机构服务与软件公司的内部测试相比，有哪些优势？

四、期待解决的问题

针对目前在测试服务方面存在的一些问题，笔者认为认清以上面提到的几个问题。

1. 软件测试服务客户

软件测试服务的发展要依托于广大的客户群。那么，哪些客户应成为软件测试服务的对象呢？目前，针对国内软件的发展状况，软件测试服务的客户主要有以下几种：

1) 大型的软件企业。

这种类型企业有着规范的软件开发过程控制，深知软件质量的重要性。在软件开发中，比较重视软件测试这个阶段，肯于花费必要的资源、资金去做质量保证工作。在这样的企业中，有专门的质量保证部门，在各开发部同时存在一定数量的测试人员。同时有着良好的过程管理模式。在测试方面它们会采取两种方式：自己的内部测试；外包给其它测试机构。

2) 中型软件企业。

中小型软件企业类型中，企业的种类比较复杂。归纳下来有以下几种：

软件开发过程比较规范。具有完善的过程管理机制，组织部门齐全。

软件开发过程规范，公司有专门的测试队伍，但缺乏合理的测试管理。

软件开发过程不规范，公司没有专门的测试队伍，测试是由开发人员或者抽调同公司别的部门的人员进行的。或者干脆不做测试。

3) 小型软件企业。

由于企业规模小，企业组织结构不完整。这样的企业基本上没有转门的测试队伍，更没有完善的测试管理机制。

企业级的软件最终使用用户。

在我国，大中型企业普及管理信息化是必然趋势。在这类企业中，各种用于企业管理的软件产品或项目数量都非常巨大。但由于这类企业没有专业的计算机质量保证人员，无法有效的验收软件产品或项目。

根据调查数据显示，目前软件测试服务的客户主要集中是大中型软件企业。而主要的合作形式为：协助本软件企业完成部分测试工作。为在质量管理薄弱的企业提供优秀的管理模式也存在很大的市场，主要客户集中在中小型软件企业。同时企业级的软件最终使用用户也是软件测试服务的重要群体。

2. 唤起客户的质量意识

随着计算机使用的普及，各行各业对计算机使用的不断提高，企业对员工的计算机水平的要求也不断提高。这样，势必会不断提高软件使用企业对软件产品的质量要求。同时，由于软件使用企业在过去的软件项目实施时，对项目验收把关不严格。在后来的使用过程中发现质量不高的软件产品为企业管理带来诸多不便。在某些行业里，因为软件的错误给企业带来的损失是巨大的。这些原因都使得软件使用企业对软件质量的要求发生变化，对软件产品的质量要求越来越高。

对于软件企业，越来越认识到软件产品的质量是企业的生命线。软件产品质量不能得到保证，会造成软件项目的失败，企业失去客户的信任，给企业带来巨大的经济损失。

软件企业和软件用户双方都对软件质量提出了高的要求。软件测试服务在这样的环境下，有着很大的发展空间和很多的商业机遇。作为软件测试服务机构应该抓住这个良好的发展机会，大力推广软件测试服务业务。

近年来，国内的软件行业内一直提倡中国的软件靠出口来发展自己。希望通过获得更多的软件外包项目来扩展生存空间。但现实的问题是，中国技术整体水平不高，软件外包讲究的是,低成本和高质量，管理比技术重要得多，而国内软件开发管理与国际先进水平相比还有一定差距。外包对软件企业管理水平、维护能力，以及商务、法律的国际接轨都有相当的要求。而我国软件企业相对松散，质量管理也处于弱势，很多还是作坊式的研发。质量管理的薄弱，使得国内的软件行业难有很强的竞争优势。现实也要求国内的软件行必须提高质量意识。

作为软件测试服务机构要广泛宣传软件质量的重要性，唤起客户的质量意思。同时提高自身业务素质和技术水平。

3. 软件测试服务结构

为了更好的服务于客户，软件测试服务结构应具有完善的管理机制；一流的计算机硬件设备；先进的软件测试工具、测试方法和测试管理体系；众多优秀的技术人材。具有丰富的软件产品项目经验，建立常见应用管理软件的测试用例库，提高软件测试的复用性，提高质量和效率。

4. 软件测试服务内容

软件测试服务机构的业务包括以下几方面：

1) 为软件企业提供第三方测试服务。
2) 为软件企业提供优秀的质量管理模式。包括：ISO、CMM、CMMMI、SJ/T11234、SJ/T11235 和测试管理模式。
3) 质量管理的咨询服务。
4) 为软件企业提供多种的测试服务：登记测试、鉴定测试、质量测试、验收测试、系统功能测试、安全测试和性能测试等。
5) 为软件使用企业提供软件项目的验收测试服务。

5. 软件测试服务的优势

专业软件测试机构具有成熟的测试流程和测试方法。可以为软件企业节约成本，包括人力资源成本和购买测试工具等。另外，专业软件测试机构专门从事测试工作，在软件测试方面具有很多知识和经验等方面的积累，更利于挖掘更多软件中的缺陷，避免思维定式。为质量管理薄弱的软件企业提供优秀的管理模式。同时为软件使用企业提供良好的技术支持。

五、小结

软件测试在国内的发展过程，同样会反映在第三方测试服务上。虽然还存在很多的问题，但软件测试和第三方测试服务将成为具有广阔发展前景的领域。软件测试和第三方测试服务的兴起意味着更多的机会。通过提供软件测试服务，国内软件本地化企业可以扩展服务业务范围，扩大企业发展规模，完善企业管理等。

浅谈软件测试自动化解决方案

张天鼎

【摘要】测试是软件开发的一个重要环节。本文论述了软件测试自动化测试的实施。从自动测试的好处. 影响软件测试自动化实施的因素产生原因等几个方面出发.总结软件自动化测试的方案。

【关键字】软件测试　软件自动化测试

软件测试自动化，已经成为国内软件工程领域一个众所周知的课题；不言而喻，软件测试从业者都意识到软件测试这项工作走向成熟化、标准化的一个必经之路就是要实施自动化测试。也许您认为实施自动化测试不是必须，也许您认为测试的思想是开展该工作的精髓、而工具只是辅助，那么我要告诉你我的想法：从计算机这一庞大学科发展至今，它最根本的意义是解决人类手工劳动的复杂性，成为替代人类某些重复性行为模式的最佳工具；我们不可推翻测试思维在测试工作中的指导思想地位，但如何将思想转化成可以具体实践的可操作性方案，那么就请君看看我这篇粗浅的论述。

以前听过北京中软的一个业内专家讲一句话，觉得挺经典：记住，凡是说既是科学又是艺术的学科，就是说明它是不成熟的学科！他将软件工程和建筑行业做类比，让我们深深体会到软件工程走向成熟化的任重与道远。而软件测试，更是一个新兴的领域，虽然近几年得到了快速发展，也随着该领域从业者数量的与日俱增，培养了一批高级的人才；但是依然有多少企业和个人工作在迷茫中：这种困惑是因为工程师们手中的测试工作与理想的测试模式造成的强烈反差，这种无奈是因为他们和开发人员一样的努力却有不同的待遇，这种迷茫是因为测试工作者不知道这个领域里是否还有自己的发展空间和人生价值的体现！笔者认为：如今的软件测试行情，正处在群雄逐鹿的混战岁月，每个人、每个有测试部门或从事测试业务的企业，都该发扬百花齐放、百家争鸣的精神，多多借鉴国内外先进的测试经验，参考业界流行的行业标准，找到适合自己团队的测试方法和模式，创造更大的社会价值，发挥更大的人生价值。这里，笔者从自身测试经验出发，参考公司的自动化测试模式，简单谈下如何实施自动化功能测试的问题，也希望同行朋友们多多补充、交流。

首先说，为什么要实施软件测试自动化呢？它的好处何在？第

一，测试人员的工作比以往任何时候都更加困难，因为公司和组织希望以更快的速度和更低的成本开发出高质量的应用程序；此外，在很多项目中，测试人员的所有任务实际上都是手动处理的，而实际上，有很大一部分重复性强的测试工作，是可以独立开来自动实现的；还有，在大型项目中测试团队和其他的团队之间没有足够的合作，无法促进彼此的工作；最后，从个人角度来说，测试人员通常很难花费大量时间来学习新技能，这是目前国内测试从业者的现状，太多的企业为了节约成本而将刚刚走出校门的毕业生作为测试工程师，他们每日做着繁忙的重复工作，又基于自身技能的不深，虽怀博览群书的心愿却不知从何出入手。所谓光阴似箭，因为一转眼我们就说到未来的 5 年、10 年后，我们这些技能不深的测试工程师能做什么呢？还是这样练鼠标熟练度吗？可以说，实施测试自动化是软件行业一个不可逆转的趋势，如果在这个领域走在了前列，无论从企业的核心竞争力还是个人的工作技能来说，都有巨大的优越性，而国内众多的软件厂商也的确在纷至沓来的着手开展着这项工作。

然而，随着很多具有了一定优秀实施自动化测试经验的企业陆续出现，也伴随着很多组织对这项工作成了丈二和尚－摸不着头脑。对当前国内软件企业实施或有意向实施测试自动化时面临的主要问题，我总结如下：

- 干脆认为测试自动化是个要不可及的事情，我们这样的小公司不必实施，人员、资金、资源都不足，以后再说吧！
- 热血沸腾的实施测试自动化，购买了工具，推行了新的测试流程；几个月后，工具放在那里成了共享资源，测试流程又涛声依旧，回到原来的模式。
- 公司实施了自动化测试；然而开发与测试之间，甚至与项目经理之间矛盾重重，出了事情不知如何追究责任；虽然还在勉强维持的自动化测试，但实施的成本比手工测试增加了，工作量比从前更大了，从而造成项目团队人员怨声载道，更怀念起那段手工测试的悠闲岁月，唉！那一场风花雪月的事，既然要结束又何必开始…
- 自动化测试实施相对比较成功，但或多或少还有些问题，比如工具选择不准确，培训不到位，文档不完备，人员分配不合理，脚本可维护度不高等等，造成一种表面上的自动化测试流程，是一幅空架子，如同山间竹笋,嘴尖皮厚腹中空。

现在让我们来简单分析下产生这些问题的原因。

1） 公司高层意识不到软件测试自动化的重要性；殊不知，其他竞争对手们都大张旗鼓的开展这方面研究和策划的时候，自己还对此持漠视态度，等到整个行业都提高到一个新的层次，那时再着手做，可能就是热锅上的蚂蚁了。

2） 所谓凡事预则立，不预则废。一个软件企业实施测试自动化，绝对不是拍脑袋说干就能干好的，它不仅涉及测试工作本身流程上、组织结构上的调整与改进，甚至也包括需求、设计、开发、维护及配置管理等其他方面的配合。后续部分会重点阐述这个问题。

3） 软件开发是团队工作，在这一领域要尤其注重一人为本；所以人员之间的配合、测试组织结构的设置非常重要，每个角色一定要将自己的责任完全担负起来，这也是减少和解决前述团队矛盾的必要手段。

4） 对开展自动化测试的监督和评估相当重要，也包括对工作产品的检查和人员的考核。一定要将自动化测试全面深入的贯彻到测试工作中，不能敷衍了事，不能做表面工作。这项工作在 CMM 三级里规范的很好，只可惜我们的很多公司对 CMM 真的只是“过级”！

下面详细谈下笔者对实施软件测试自动化的粗略见解。

这里我会以普通测试工程师的角度，先谈下针对第一种情况的公司，如何实施自动化测试。

如果您所在的公司至今还没有做自动化测试或做自动化测试的意向，而其原因是因为公司高层对此不屑一顾，或者领导认为公司规模小、资金不足等原因，那么这的确是个棘手的问题，因为这不是我们普通工程师能够力量所及的。针对这种情况，我想凭借曾经的经验来说下个人的体会。我从前就是工作在这样的一个环境里，曾千方百计引导公司领导，希望对测试部门和测试自动化引起重视，但是我失败了。面对 50 来个开发人员对应 6、7 名测试人员的严重不协调比例，面对项目开发周期的一再缩短而最终都以更大缩短测试时间的巨大代价，面对我们从测试角度看根本不能交付的版本去给用户上线的软件，面对测试人员付出和其他项目成员同样甚至更多的劳动后、待遇却是人家的一半或 n 分之一，哎哟，我这心哪——真如赵本山所言——瓦凉瓦凉的！在向上级汇报我的任何想法都无济于事的时候，所谓痛定思痛，不能破罐子破摔，我们要换个思维方式来看问题。从根本上说，在我们个人还没有到达可以在公司中一言九鼎的地步，那么首先

要做的，就是练就个人的本事和技能。从测试角度讲，我们认识到了软件测试行业发展的广阔空间，我们知道软件测试自动化的重要性，在这方面我们的思想是先进的；那么何不自己全力提高在这方面的能力，然后带动更多的人来一齐推动这件事，那时候我们的意见就会或多或少得到高层的重视了。即便公司领导顽固不化，我们也不会怀才不遇，此处不留人，自有留人处！在经过一年的知识储备和技能提高后，一个偶然的机会我被一个至今我相当感激的好朋友介绍到另个公司，继续从事着我的自动化测试工作。这里，我要说明几点其他体会：

目前国内的软件公司，很多还是处于获取资本的原始积累阶段，我们不能说公司领导完全不重视测试，而是测试整体行业都没有被重视起来，这是其一；其二是公司高层有更需要重视的环节，例如寻找客户签订单，或者开发，这些是直接关系公司存亡的。

即便企业重视测试，如果公司做一番比较全面的评估，也不一定非要实施自动化测试。在后续的测试自动化切入条件里，再详细说明。公司可以在测试流程管理、测试缺陷流程、测试人员技能培训等方面做工作，这样可以用比较少的成本投入来获取相对较大的回报。我们都知道如果真的购买一款测试工具，价格的确不悱，这是很多中小企业难以承受的。

如果公司有实施自动化测试的意向，一个出色的的测试主管或自动化技术专家是必要的。首先他和高层沟通时，比普通工程师有分量；其次他对自动化测试的深刻认识能给领导提出一些建设性的意见。我们要努力成为这样的人。

最后，如果你只是一个普通测试工程师，那么在这样的公司里，你也有很多意想不到的优越性。首先，这样的公司测试不正规，那么你有更大更多的发挥空间；其次，你可以单独学习自动化测试技术，先自己试着应用到日常工作的项目中；再次，你可以直接和开发人员沟通，甚至向他们学习开发技术，这对将来的自动化测试中开发脚本很有好处；最后，每个工程师的成长都有个过程，一般来说，这样的公司喜欢招收初出茅庐的毕业生，所以这样的机会更应该被抓住，在这样的环境里练就两年的本领，出去以后，会别有一番蓝图让你勾勒。我也遇到很多和我当时一样的测试新手向我发牢骚，很惭愧我没有什么可以告诉他们的，这里只是希望这样的新人克服浮躁情绪，稳扎稳打练好基本功；想成为测试主管或测试专家，就要经过这个过程。

接着阐述第二个论题。软件企业有意向实施自动化测试，那么应该具备什么样的条件才可以引入自动化测试呢？才可以最大可能的

减少引入风险，并能够可持续性的开展下去？

首先，从企业规模上来说，没有严格限制。无论公司大小，都需要提高测试效率，希望测试工作标准化，测试流程正规化，测试代码重用化；所以第一要做到的，就是企业从高层 CTO 开始，直到测试部门的任何一个普通工程师，都要树立实施自动化测试的坚定决心，不能抱着试试看的态度。一般来说，一个这样的软件开发团队可以优先开展自动化测试工作：测试－开发人员比例合适，比如 1：1 到 1：1.5；开发团队总人数不少于 10 个。当然，如果你的公司只有三五个测试人员，要实施自动化测试绝非易事；不过可以先让一个、两个测试带头人首先试着开展这个工作，不断总结、不断提高，并和层层上司经常汇报工作的开展情况，再最终决定是否全面推行此事。

其次，从公司的产品特征来说，一般开发产品的公司实施自动化测试要比开发项目的公司要优越些。原因很简单，就是测试维护成本和风险都小。产品软件开发周期长，需求相对稳定，测试人员可以有比较充裕的时间去设计测试方案和开发测试脚本；而项目软件面向单客户，需求难以一次性统一，变更频繁，对开发、维护测试脚本危害很大，出现问题时一般都以开发代码为主，很难照顾到测试代码。但决不是说做项目软件的公司不能实施自动化测试，当前国内做项目的软件公司居多，有很多正在推行 CMM 等级标准，这是好事情；只要软件的开发流程、测试流程、缺陷管理流程规范了，推行自动化测试自然水到渠成。

接着，说说标准化的开发和管理流程。不管是 CMM 还是 ISO，不管是开发流程、测试流程还是缺陷管理流程，我这里不能一一阐述，可以参考 rup 过程，可以参考很多业界文献，我只说明一点，也是我们 IT 从业人员甚至任何从业人员一个很好的工作原则：a、把你想做的写下来（计划管理）b、按照你写下来的去做（行为管理）c、把做的事情记录下来（报告管理）d、出现的问题要设法解决（跟踪管理）。在测试流程里，这几个要点都一一有所落实；如果你的软件开发团队据此开发软件，那么完全具备实施自动化测试的条件。当然，也许一些公司的测试管理比较混乱，出了问题不知道谁负责，测试人员或开发人员整日碌碌却无为，软件缺陷不胜枚举，那么个人觉得还是首先从管理角度来规范一下公司的开发流程和测试流程吧！

最后，从测试人员个人素质和角色分配来说，除了有一个 CTO 级人物做后盾外，还应该有个具有良好自动化测试背景和丰富自动化测试经验的测试主管，不光在技术方面，更重要的是在今后的自动化

测试管理位置起着领导的作用。还要有几个出色的开发经验良好的测试人员，当然也可以是开发工程师，负责编写测试脚本、开发测试框架；如果可以，开发企业自己的测试工具；他们不需要对产品业务了解深刻，但要具有将软件业务逻辑转化成可测试逻辑的分析能力，属于自动化测试设计者。还有一些测试执行者，他们要对软件产品业务逻辑相当熟练，配合测试设计者完成设计工作，并在执行自动测试时，敏锐的分析和判断软件缺陷。如果你的测试团队具有这样的人员角色雏形，那么具备了实施自动化测试的又一条件。

综合分析上述四个条件，企业可以决定是否推行自动化测试；但是为了减少实施风险，我们还要做综合的自动化测试引入评估。

进行正确的评估手段是非常重要的，这里从以下几方面说明：

其一是资金评估。虽然你的公司具备实施自动化测试的条件，但如果企业效益不好，还是先扭亏为盈吧。一款正版的测试工具价格庞大，企业要首先考虑资金是否允许购买正版的测试工具软件，所以进行测试工具的成本估算，以及引入自动化测试后组织结构调整等方面的成本估算是很必要的。如果你的公司处在如同前面所言的自动化测试试验阶段，可以使用试用版测试工具。当然具有实力的公司可以按照自身的工作流程自主开发测试工具，本文不考虑这种情况。

其二是企业开发的产品业务和功能是否需要自动化测试，包括白盒自动化测试、功能自动化测试和性能自动化测试。比如一些公司开发单机版软件，只需要做功能测试，那便不必考虑第三种；有的公司开发简单界面之类的软件，例如搜索引擎，也可不必考虑第二种；而大多数国内公司开发的软件，由于各种原因，一般都不考虑第一种。也有可能公司开发的软件特殊性很强，市场上根本没有支持它的自动化测试工具，此时要另辟蹊径。这种评估相当重要，要根据自身的产品功能特征来综合评估。

其三是企业软件所应用的开发语言。当前业界流行的测试工具有几十种，相同功能的测试工具所支持的环境和语言各不相同，这里笔者总结了当前国际上流行的各种自动化测试工具名称、功能简述、支持环境和生产厂商。

其四…

其五还要做时间估算。时间估算归根结底属于资金评估里成本估算的范畴，在评估完前面几项指标后，需要估算实施测试自动化的时间周期，以防止浪费不必要的时间，减少在人员、资金、资源投入上的无端消耗。虽然到测试自动化步入正轨以后，会起到事半功倍的效

果，但前期的投入巨大，要全面考虑各种因素，明确实施计划并按计划严格执行，才能最大限度降低风险。

智能测试自动化

——基于应用程序行为的模型驱动测试方法

作者：Harry Robinson
译者：江上舟

【摘要】如何提高测试的效率，如何让测试人员在测试过程中不会感到单调乏味，是我们一直在思考的问题。本文通过一个虚构的故事，提出了一种根据应用程序的行为描述来生成测试的模型驱动方法，并同手动测试、静态自动化测试和随机测试进行了对照，希望能给人以启示。

【关键字】模型驱动测试 自动化测试 状态模型

快速浏览

- 通过建模来提高你的自动化测试的效率。
- 克服手动测试和静态自动化测试的局限性。

注意：你将要读到的故事是虚构的——它很短小，但其寓意是真实的。

在产品周期中，有四位测试人员根据要求开始测试软件。

测试员 1

立即开始手动测试，并发现一些细微的错误。开发团队高兴的修复了这些错误，然后提供一个新的软件版本以供测试。测试的越多，发现的错误越多，修复的错误也就越多。

测试员 1 觉得很有成就感，也就会感到快乐——至少一段时间是这样的。

经过几轮这种发现、修复的循环，他开始由于一遍遍的手动重复运行实质上一样的测试而感到乏味和反应迟钝。当测试员 1 最终丧失积极性——同时也就意味着失去耐性——就会宣称软件可以发布了。

用户发现它有太多的错误，于是购买了竞争者的产品。

测试员 2

从手动测试开始，但很快就判定创建自动执行按键的测试脚本更

有意义。仔细找出那些会使用到软件有用部分的测试后，测试员 2 将操作记录到脚本中。这些脚本很快达到几百个。按下一个按钮后，这些脚本就被激活并按照步骤运行软件。

测试员 2 觉得自己很聪明，也就会感到快乐——至少一段时间是这样的。

当软件发生变化时，这些脚本需要大量的维护。他花费数个星期和开发人员争论，要求停止修改软件，因为这破坏了自动化测试。最后，脚本需要太多的维护以致留下太少的时间来进行测试。

当软件发布后，用户发现太多脚本未覆盖的错误。他们停止购买该产品而决定等待版本 2 的发布。

测试员 3

不想维护数以百计的自动化测试脚本。她编写了一个测试程序来在应用程序中到处随机点击和按按钮。这种“随机”测试程序不需要一直查看，且发现了很多致命的错误。

测试员 3 很享受发现这些引人注目的缺陷，也就会感到快乐－－至少一段时间是这样。

由于随机测试程序只能发现那些毁坏应用程序的错误，因此测试员 3 仍然不得不做大量手动测试，并很快在这个过程中感到乏味和反应迟钝。当软件发布后用户在软件中发现如此多的功能性错误而对公司丧失信任，于是停止购买这种软件。

测试员 4

通过从手动测试、探索式测试开始熟悉了应用程序——用手动测试中所获得的知识来为应用程序创建一个非常简单的行为模型。测试员 4 接着使用一个测试程序来测试应用程序的行为是否和模型预测的一致。行为模型相比被测的应用程序要简单的多，因此它很容易创建。由于测试程序知道应用程序应该怎样做，因此当应用程序犯了错误时它能够检测到。

随着产品周期的发展，开发人员为应用程序写出了新的特性。测试员 4 很快更新了模型，测试得以继续进行。程序白天和晚上都可以运行，持续生成新的测试序列。测试员 4 可以一次在多台机器上执行测试，将多天的测试放到一个晚上来完成。

经过几轮测试和错误修复后，测试员 4 的测试生成程序开始发现很少的错误。测试员 4 为了测试增加的行为更新了模型，然后继续测

试。测试 4 也会为应用程序中那些不值得建模的部分做一些手动的测试和静态自动化测试。

当测试员 4 的软件发布出来后，只有非常少的错误能被发现了。用户感到高兴。投资者感到高兴。

测试员 4 感到高兴。

评注

这四个场景展现出了当前软件测试中可用的一些方法。

测试员 1 是一个典型的手动测试者，从键盘手动运行所有的测试。手动测试在当前的工业界很普遍——它能提供直接的好处，但长时间的运行会让测试人员感到单调乏味，对公司来讲成本高。

“我看到的最悲哀的景象之一就是一个人在键盘上手动操作一些可以自动运行的东西。这是悲哀的但也是有趣的。”

——Boris Beizer，黑盒测试：软件和系统功能测试技术

测试员 2 实践的是我称为静态测试自动化的测试。静态自动化脚本每次根据同样的次序执行同样的命令序列。当应用程序发生变化时这些脚本的维护成本很高。测试是不断重复的；但由于它们总是执行相同的命令，因此它们很难发现新的错误。

“高度重复的测试实际上将发现所有重要问题的几率最小化了，这和沿着别人的足迹前行将发现自己的天地的几率最小化的原因是一样的”

——James Bach，“骗人的测试自动化,”Windows 技术期刊,1996.10

测试员 3 的操作接近于自动化测试的边缘。这些类型的随机测试程序被称为蠢猴因为它们就是毫无目的的敲打键盘。它们生成非常规的测试执行序列并发现很多致命的错误，但是想控制它们到应用程序中你想测试的部分却是很困难的。因为它们不知道自己在做什么，所以它们会漏过应用程序中很多明显的错误。

“猴子式的测试不能是你测试的全部。猴子不明白你的应用程序，由于它们的无知它们漏掉了很多错误。”

——Noel Nyman，“使用猴子式的测试工具,”SQTE,2004.1/2

测试员 4 通过一种称为“模型驱动测试”的智能测试自动化方法糅合了其它测试员的方法。

模型驱动测试并不像静态测试自动化那样逐字逐句的记录测试序列，也不盲目的在键盘上敲打。模型驱动测试通过对应用行为的描述来判断哪些操作是可能的、期望输出是什么。这种方式不断自动生

成新的测试序列，很好的适应了应用程序的变化，能够同时在多台机器上运行，并能整天运行。

“一个艺术家用他的智慧画画，而不是用他的手。”

——Michelangelo Buonarroti

这个故事的寓意

手动测试是开始测试自动化过程的好的方法。我把这个阶段称为“探索式建模”，因为它综合了探索式测试过程和用来生成测试的模型的发现过程。当你开始搞清每种操作的行为后，你就能创建能帮助建模和测试应用程序的规则。

测试员 1 的方法需要他的手不停的在键盘上工作。最后测试员 1 精疲力竭。

测试员 2 的静态脚本重复他的手已经执行过的那些键盘操作。

测试员 3 的猴子式测试本质上是无目的的在键盘上乱敲。

测试员 4,从另一方面，在其它技术上进行了补充：

- 思考应用程序的行为，
- 将行为描述给一个测试生成程序，
- 让测试生成程序来创建和运行测试用例。

通过根据应用程序行为描述生成测试，测试员 4 能够执行那些在使用其它测试方法时不可实现的测试。

这个故事的寓意：自动化你的大脑，而不只是你的手。

使用你的大脑

让我们看一个创建和使用行为模型来测试应用软件的例子。

手动测试是开始测试自动化过程的好的方法。我把这个阶段称为“探索式建模”，因为它综合了探索式测试过程和用来生成测试的模型的发现过程。当你开始搞清每种操作的行为后，你就能创建能帮助建模和测试应用程序的规则。

这就是模型驱动测试的精髓：按照一种能够被用来生成测试的方式来描述行为。针对你将要测试的每一种操作，问自己以下两个问题：

1. 什么时候这种操作是可能的？
2. 当这种操作被执行时输出是什么？

例如，假设你被要求测试 Windows 目录下文件的行为。更确切一点，你要测试创建、删除和反选取操作。

为创建操作建模

- 什么时候创建是可能的？为了简单，在这个例子中限制目录中的文件数为 1.这样只有在目录中有 0 个文件时可以创建。
- 当创建被执行时输出是什么？当你在一个目录中创建一个新的文件时，这个目录中的文件个数加 1。这个新创建的文件默认是被选中的，因此在目录中它是加亮的。实际上这个新文件是这个目录中唯一被选的文件，不管在创建操作前有多少被选中。

为删除操作建模

- 什么时候删除是可能的？只有当目录中至少有一个被选中的文件时删除才是可能的。
- 当删除被执行时输出是什么？当你执行删除操作时，目录中任何被选中的文件都将消失。

为反选取建模

- 什么时候反选取是可能的？在这个模型中反选取总是可能的，即使目录中有 0 个文件。
- 当反选取被执行时输出是什么？当你执行反选取操作时，目录中所有被选取的文件将都不再被选取，而所有没被选取的文件将被选取。当目录中有 0 个文件时，反选取让目录保持不变。

建立一个状态模型

如图 1 所示，现在你可以构建一个系统行为的“状态模型”。它将上面描述的那些行为合在一起。注意反选取操作从 0 文件状态回到 0 文件状态的方式。它模拟出当没有什么需要反选取时什么都不做的情况。

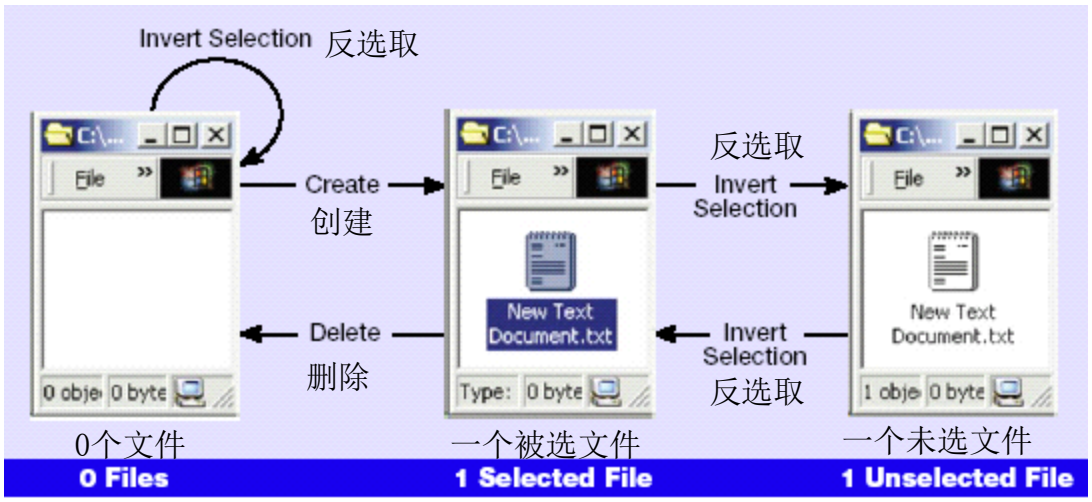


图 1 状态模型

很好，那接下来呢？

现在你明白应用程序是如何工作的，那么你就可以手动测试这些操作、检验 Windows 目录是否按照你的预期变化。但是，由于你的理解都存在于你的大脑中，你的测试结果也就受你的时间和体力所限。

另一方面，如果你能将这个状态模型从你的大脑完全的移植到计算机上，计算机将能替你为系统生成和执行测试。

幸运的是，这个模型可以通过计算机能识别的被称为“状态表”的形式进行表达。状态表（见表 1）的每一行表明某种操作用于处于开始状态的应用程序后会到达的结束状态。

开始状态	操作	结束状态
0 个文件	反选取	0 个文件
0 个文件	创建	1 个被选文件
1 个被选文件	反选取	1 个未选文件
1 个被选文件	删除	0 个文件
1 个未选文件	反选取	1 个被选文件

表 1 针对 Windows 目录中文件行为的状态表

也使用计算机的大脑

一旦我们将状态模型转换成计算机能理解的状态表，计算机可以为我们做些什么呢？我们该如何使用我们有关应用程序行为的那些信息呢？

计算机能够应用状态表来生成测试序列以测试应用。你将在下面的例子中看到，可根据这些测试序列的新颖性、有效性或它们的全面性来选择它们。这种测试生成是应用你对系统的理解的一种强有力的方式——这就是模型驱动测试的内容。

状态模型上的随机路线

生成测试操作的一个简单的方法就是从应用当前状态可用的操作中任意选择一个。例如，如果你在 0 个文件开始状态，你能够选择下面两种操作中的任意一个：

- 反选取（让你仍停留在 0 个文件状态）
- 创建（让你停留在 1 个被选文件状态）

按照这种方式随机选取操作，你能生成许多非常规的序列（就象测试员 3 的随机“猴子测试程序”那样），而最后你将能执行模型中的所有操作。图 2 显示了一个典型的随机路线。注意这个随机路线在一轮中执行了四次同样的操作（反选取），但是却让剩下的两个其它的操作没有执行到。这就是随机测试的实际情况。

图 2 一个随机路线

	执行的操作	结束状态
1	创建	1 个被选文件
2	反选取	1 个未选文件
3	反选取	1 个被选文件
4	反选取	1 个未选文件
5	反选取	1 个被选文件

状态模型上一个有效的路线：中国邮递员路线

当模型很大时要到达所有测试操作按照随机路线是很低效的。我们怎样才能有效的测试模型中的每种操作呢？

一个邮递员在递送信件时会遇到同样的问题。设想模型中的每个操作是递送邮件必须经过的街道——模型中每个状态是邮递员改变方向时的十字路口。就像邮递员必须走过每一条街道来递送邮件一样，我们也必须测试模型中的每种操作。在这两种情况下，我们都想将所需的路线长度减小到最短。

一个叫做管梅谷的中国数学家为这个问题提出了一个优秀的解决方案，这就是中国邮递员算法（见图 3）。管梅谷的方法生成一条该状态模型的路线，该路线用最少的步数执行到模型中所有的操作。下面列出的测试序列只用 5 步就覆盖了模型中的 5 种操作。对于一个你要快速测试的、大的应用程序而言这种效率是很有用的。

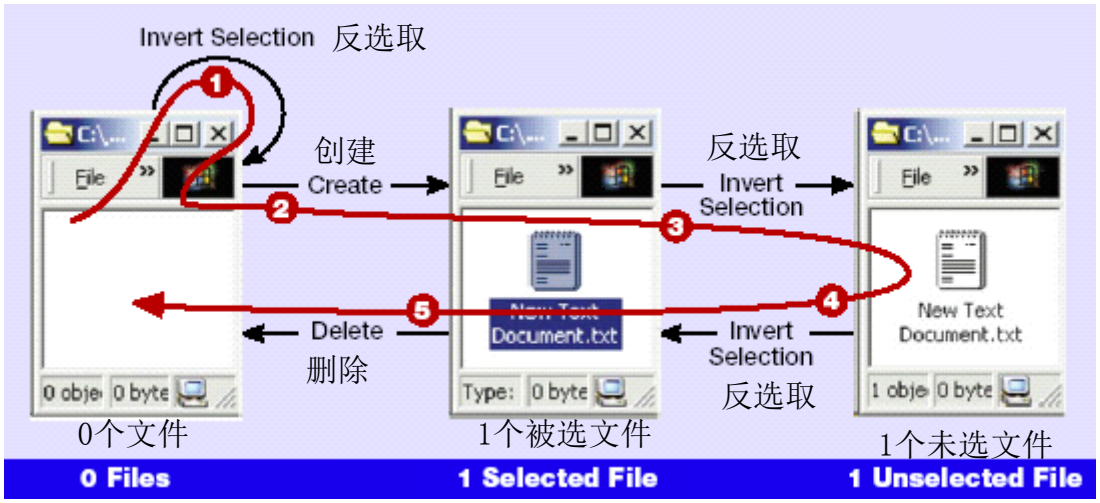


图 3 一个中国邮递员路线

	执行的操作	结束状态
1	反选取	0 个文件
2	创建	1 个被选文件
3	反选取	1 个未选文件
4	反选取	1 个被选文件
5	删除	0 个文件

一个更有效的路线：状态改变的中国邮递员路线

模型中的一些操作——如当目录中有 0 个文件时点击反选取——不会改变应用程序的状态。如果你认为当应用程序的状态发生变化时

更容易出现错误，你可以优先测试状态改变的操作。

一个简单的方法是将不会改变状态的操作从状态表中过滤掉。对于表 1，需要移除第一个操作（反选取）。

在这个简化的状态模型上运行中国邮递员算法生成一个用四步覆盖模型中所有状态改变的操作的测试序列——本质上就是将前一个例子中的第一步移除：

	执行的操作	结束状态
1	创建	1 个被选文件
2	反选取	1 个未选文件
3	反选取	1 个被选文件
4	删除	0 个文件

回到开始状态的最短路线

假设你想全面的测试所有让 Windows 目录从 0 个文件状态经过几步回到 0 个文件状态的测试序列。这种不断产生新的变化的测试序列对于测试员 2 的静态自动化而言是不可想象的。

对于计算机而言根据状态模型生成一系列这种路径是微不足道的。你可以生成长度不断增加的测试序列直到让计算机死掉，这样就能越来越深入的探索模型。

路径 A 有一步：

A1：反选取

路径 B 有两步：

B1：创建

B2：删除

路径 C 有四步：

C1：创建

C2：反选取

C3：反选取

C4：删除

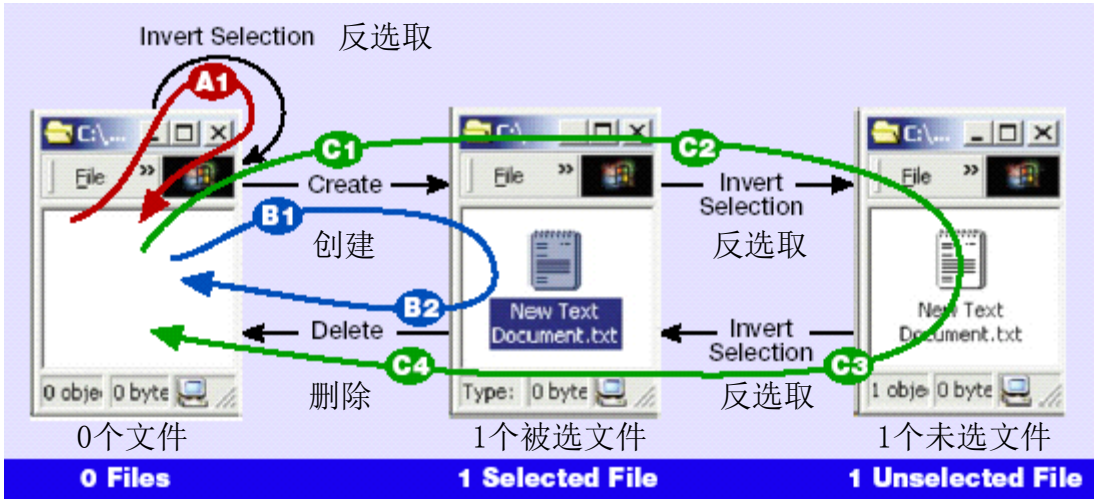


图 4 四步或更少的状态回环

使用计算机的手

每种算法的输出是需要执行的测试操作的序列。哪种是执行这些操作的最佳方法？你可以将操作列表交给一个测试人员去手动执行——但这肯定是缓慢的、单调乏味的和痛苦的。谁愿意把他们的时间都花费在执行这些操作上。这种重复性的工作就是那种让测试员 1 如此痛苦的罪魁祸首。

作为替换，你可以编写一个简单的测试执行程序来读取列表并执行针对列表中每种操作的测试代码。

例如，使用 Visual Test，创建操作的执行测试代码为：

```
WToolbarButtonClick("@1","File") ' Open the File menu
WMenuSelect("New") ' Select New File
WMenuSelect("Text Document") ' Choose Text Document
Play "{Enter}" ' Accept the default filename
```

在典型的静态自动化中，这段代码将嵌入脚本中——但在一个模型驱动测试程序中，这一小段代码将只在列表中的测试操作需要执行创建操作时才会被调用。

使用计算机的眼睛

自动化测试操作只是战斗的一半。你还需要一种自动判断应用程序是否正确工作的方法。

这种方法——一种用来判断应用程序是否正确响应测试操作的方法——被称为测试准则。一些测试方法，如测试员 3 的随机猴子式测试程序，必须基于类似于应用程序是否崩溃的痛苦的测试准则。

模型驱动测试让测试程序有能力检查那些比“系统不崩溃”更细的行为指标。根据状态表中的信息，模型知道每一个状态下哪些操作是可用的以及每种操作的期望输出。例如，模型指出测试程序在一个被选文件的状态下可以执行删除操作。模型同时也指出执行删除操作的结果就是让应用进入 0 个文件状态。这些知识提供了两种检验应用程序是否正确运行的方法。

第一，测试程序可以判断是否能执行的操作不能执行。如果在应用处于 1 个被选文件的状态下不能执行删除操作，测试程序将上报一个错误，因为在无法找到菜单中的可选删除项时测试程序会失败。

第二，模型始终知道应用程序应该进入哪种状态。知道每种操作的期望结束状态也就意味着我们能够创建测试准则程序来检查（在每种操作结束后）目录中是否有合适的文件数和被选文件数。例如，当上述的删除操作被执行后，结束状态就应该是目录中有 0 个文件（当然也就是 0 个文件被选）。

程序化的测试语言通常提供可用于测试程序检查应用程序各个方面的函数。两个对当前模型有用的 Visual Test 函数是：

WviewCount()指出目录中的文件数，

WviewItemSelected()指出目录中有多少文件被选。

测试程序能够验证应用程序是否处在正确的状态，如表 2 所示。

应用程序的状态	WviewCount 的期望返回值	WviewItemSelected 的期望返回值
0 个文件	0	0
1 个被选文件	1	1
1 个未选文件	1	0

表 2 显示 Visual Test 函数 WviewCount()和 WviewItemSelected()的状态表

上面讨论的删除操作将让应用进入 0 个文件状态。如果 WviewCount()返回一个非 0 的值，测试程序准则将上报一个错误，因为目录中的文件数不正确。

如何更新模型驱动测试

还记得测试员 2 的静态测试自动化尝试由于应用程序的改变而让人灰心吗？相反的是，测试员 4 能够利用模型驱动测试自动化来很快适应应用程序的改变。

将新的操作合并到模型中

假设你的开发团队告诉你他们实现了全选操作。你该怎样为新的操作更新你的测试呢？很简单——升级你的状态模型来并入全选操作，然后重新生成测试。

第一，通过回答我们那两个基本问题来为全选建模：

1. 什么时候全选是可能的？在这个模型中全选总是可能的，即使目录中只有 0 个文件。
2. 当全选被执行时输出是什么？当你执行全选时，目录中的所有文件变成了被选。如果目录中有 0 个文件，全选将使得目录没有变化。这种情况在下面的图示中指示了出来，全选操作从 0 个文件状态回到 0 个文件状态。

图 5 中显示了合并全选之后的新的状态模型。

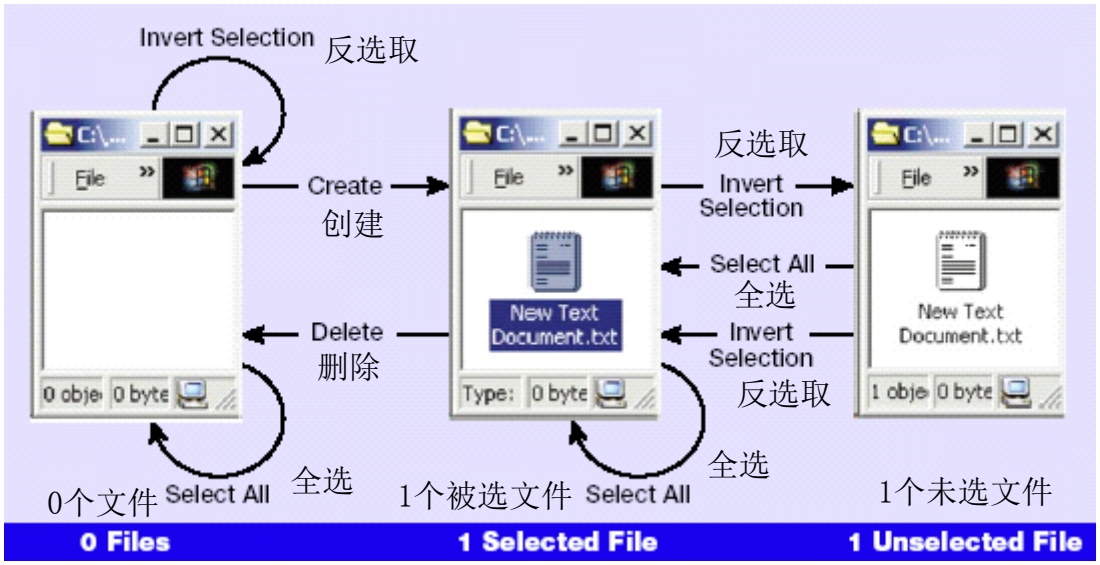


图 5 包含全选的状态模型

在升级后的模型上（见图 6）运行中国邮递员算法得到了一个九步的测试序列——用 0 个文件状态作为开始状态——执行了模型中所有操作，其中包括新加的全选操作：

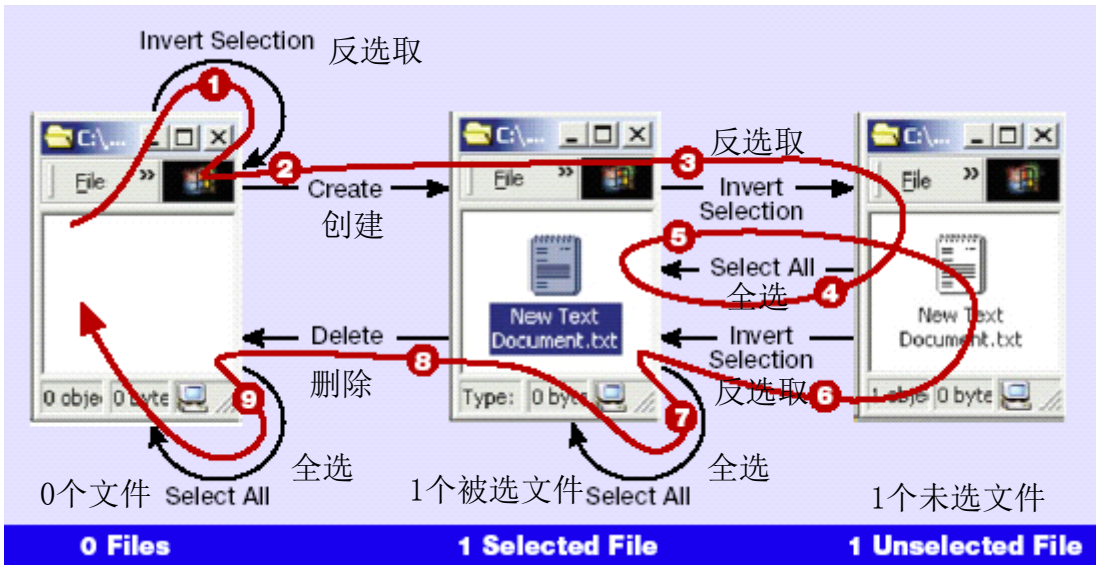


图 6 新状态模型下的中国邮递员路线

	执行的操作	结束状态
1	反选取	0 个文件
2	创建	1 个被选文件
3	反选取	1 个未选文件
4	全选	1 个被选文件
5	反选取	1 个未选文件
6	反选取	1 个被选文件
7	全选	1 个被选文件
8	删除	0 个文件
9	全选	0 个文件

下一步就是确定用于调用全选操作的代码，不管该操作在测试序列中何时发生。使用 Visual Test 该代码如下：

```
WtoolButtonClick("@1","EDIT")
WmenuSelect("Select All")
```

总结

弄清应用程序并为其建模需要相当大的努力。放弃路线简单的手动测试和路线足够长的静态自动化测试，而花费时间思考如何对应用程序进行测试是困难的——就象我们在虚构的故事中所看到的四位测试员所进行的试验和经历的磨难那样。

但是我们获得的回报是丰厚的：

- 模型驱动测试几乎从开发的第一天起就开始创建灵活的、有

用的测试自动化。

- 模型的修改很简单，因此模型驱动测试在项目的生命周期中进行维护是很经济的。
- 模型能够根据你的需要来生成数不清的测试序列。
- 模型让你能够在短时间内完成更多的测试，因为一个测试生成程序能整天在多台机器上创建和验证测试序列。
- 模型驱动测试能对其它形式的测试进行补充，能够执行那些在其它测试方法下不可实现的测试。

你和我都知道软件测试不是虚构的故事，不可能保证整个过程中都是愉快的。但是在你的测试中加入模型驱动的智能将是帮助你找到通向快乐的终点的强大的工具。

作者简介

Harry Robinson 是微软智能搜索组的软件测试经理。他维护着模型驱动测试的主页（www.model-based-testing.org）,是模型驱动测试长时间的倡导者和开拓者。

软件测试园地

——摘自 51Testing 论坛每日一贴

（www.51testing.com）

软件测试自动化的概念

软件测试自动化，是一项让计算机代替测试人员进行软件测试的技术。他可以让测试人员从繁琐和重复的测试活动中解脱出来，专心从事有意义的测试设计等活动。如果采用自动比较技术，还可以自动完成测试用例执行结果的判断，从而避免人工比对存在的疏漏问题。设计良好的自动化测试，在某些情况下可以实现“夜间测试”和“无人测试”。在大多数情况下，软件测试自动化可以减少开支，增加有限时间内可执行的测试，在执行相同数量测试时节约测试时间。

软件测试自动化通常借助测试工具进行。测试工具可以进行部分的测试设计、实现、执行和比较的工作。通过运用测试工具，可以达到提高测试效率的目的。所以测试工具的选择和推广使用应该给予重视。部分的测试工具可以实现测试用例的自动生成，但通常的工作方式为人工设计测试用例，使用工具进行用例的执行和比较。

软件测试自动化的设计并不能由工具来完成，必须由测试人员进行手工设计，但是在设计是却必须考虑自动化的特殊要求，否则无法实现利用工具进行用例的自动执行。为此，就必须在测试的设计和内容的组织方面采取一些特殊的方法。

对于软件测试自动化的工作，大多数人都认为是一件非常容易的事。其实，软件测试自动化的工作量非常大，而且也并不是在任何情况下都适用，同时软件测试自动化的设计并不比程序设计简单。

自动化测试的优点

通过自动化测试，可以使某些任务提高执行效率。除此之外，还有其它优点：

①对程序的回归测试更方便。这可能是自动化测试最主要的任务，特别是在程序修改比较频繁时，效果是非常明显的。由于回归测试的动作和用例是完全设计好的，测试期望的结果也是完全可以预料的，将回归测试自动运行，可以极大提高测试效率，缩短回归测试时

间。

②可以运行更多更繁琐的测试。自动化的一个明显的好处是可以在较少的时间内运行更多的测试。

③可以执行一些手工测试困难或不可能进行的测试。比如，对于大量用户的测试，不可能同时让足够多的测试人员同时进行测试，但是却可以通过自动化测试模拟同时有许多用户，从而达到测试的目的。

④更好地利用资源。将繁琐的任务自动化，可以提高准确性和测试人员的积极性，将测试技术人员解脱出来投入更多精力设计更好的测试用例。有些测试不适合于自动测试，仅适合于手工测试，将可自动测试的测试自动化后，可以让测试人员专注于手工测试部分，提高手工测试的效率。

⑤测试具有一致性和可重复性。由于测试是自动执行的，每次测试的结果和执行的内容的一致性是可以得到保障的，从而达到测试的可重复的效果。

⑥测试的复用性。由于自动测试通常采用脚本技术，这样就有可能只需要做少量的甚至不做修改，实现在不同的测试过程中使用相同的用例。

⑦可以让产品更快面向市场。自动化测试可以缩短测试时间，加快产品开发周期。

⑧增加软件信任度。由于测试是自动执行的，所以不存在执行过程中的疏忽和错误，完全取决于测试的设计质量。一旦软件通过了强有力的自动测试后，软件的信任度自然会增加。

总之，通过较少的开销获得更彻底的测试，提高软件质量，这是测试自动化的最终目的。

自动化测试中常见的问题

在软件测试自动化的实施过程中会遇到许多问题，以下是一些比较普遍的问题：

①不现实的期望。一般来说，业界对于任何新技术的解决方案都深信不疑，认为可以解决面临的所有问题，对于测试工具也不例外。但事实上，如果期望不现实，无论测试工具如何，都满足不了期望。

②缺乏测试实践经验。如果缺乏测试的实践经验，测试组织差，文档较少或不一致，测试发现缺陷的能力较差，那么首先要做的是改进测试的有效性，而不是改进测试效率。只有手工测试积累到一定程

度，才能做好自动化测试。

③期望自动测试发现大量的新缺陷。测试第一次运行时最有可能发现缺陷。如果测试已经运行，再次运行相同的测试发现新缺陷的概率就小得多。对回归测试而言，再次运行相同的测试只是确保修改是正确的，并不能发现新的问题。

④安全性错觉。如果自动测试过程没有发现任何缺陷，并不意味着软件没有缺陷。可能由于测试设计的原因导致测试本身就有缺陷。

⑤自动测试的维护性。当软件修改后，通常也需要修改部分测试，这样必然导致对自动化测试的修改。在进行自动化测试的设计和实现时，需要注意这个问题，防止自动化测试带来的好处被高维护成本所淹没。

⑥技术问题。商业的测试工具也是软件产品，并不能解决所有问题，通常在某些地方还会有欠缺。测试工具都有适用范围，要很好的利用它，对使用者进行培训是必不可少的。

⑦组织问题。自动测试实施并不简单，必须有管理支持及组织艺术。

测试自动化的限制

测试自动化可以带来非常明显的收益，但也有其限制，主要有：

1.不能取代手工测试

2.手工测试比自动测试发现的缺陷更多

3.对测试质量的依赖性极大

4.测试自动化不能提高有效性

5.测试自动化可能会制约软件开发。由于自动测试比手动测试更脆弱，所以维护会受到限制，从而制约软件的开发。

6.工具本身并无想像力

另外，人工测试比测试工具更优越的另一个方面是可以处理意外事件。虽然工具也能处理部分异常事件，但是对真正的突发事件和不能由软件解决的问题就无能为力。

什么是应首先被自动化的测试？

软件测试的自动化过程是一个渐进的过程，并不需要一开始就对所有的测试进行自动化，这通常也是不现实的。如何选择首先被自动化的测试成了最先遇到的问题。

有些测试，完全没有必要进行自动化，因为自动化它们所需的时间比手工运行它们全部的次数所需的时间总和还长。例如，手工运行一个测试要 10 分钟，而且一般每个月运行 1 次，那么一年需要 120 分钟。但如果自动化这测试需要 10 小时，那么这个测试需要连续不断运行 5 年才能收回成本。

有些测试，虽然执行的时间不长，但过程繁琐，需要执行的动作非常多。比　如，一个运行 10 分钟的测试，可能需要击键 150 次，打开 4～5 个窗口，切换操作。如果将其自动化，可以提高可靠性，也是值得的。

对软件进行的功能性测试，是测试软件系统在做什么。这些测试可以明确的知道应该在什么情况下输入什么，会有什么样的输出。这样的测试是非常容易被自动化的，也能从自动化中获得较大的收益。

对软件进行的性能测试，包括在不同的系统负载下进行的测试。这些测试需要采用工具辅助完成，非常适合进行自动化。

如果在测试中，运行 10%的测试需要花费 90%的时间，那么将这 10%的测试自动化是值得的。

以下列出了选择首先进行自动化时要考虑的因素：

1) 非常重要的测试
2) 涉及范围很广的测试
3) 对主要功能的测试
4) 容易自动化的测试
5) 很快有回报的测试
6) 运行最频繁的测试

应该注意避免一口气自动化太多的测试。太多的工作导致参与人员工作积极性下降，可维护性下降，增加工作的风险。寻找可快速制胜的测试，尽快让大家看到工作成果，有助于获得更多的工作支持。

工具的选择：创建还是购买

在评估了商业市场后，你可能会发现在你的限制之内没有符合你需求的工具。这时需要考虑是否自行开发自己的工具，还是等待市场上出现满足要求的新工具。

自行开发新工具有以下特点：

1、它将是最合适你的需求的
2、可以在工具中补偿被测软件缺乏的可测试性
3、工具可以假设很了解被测程序，因而减少了实现测试自动

化所需的工作

4、在文档、帮助和培训方面可以不用提供很好的支持

5、工具可能具有某些典型的问题，如结构、可扩展性等

6、用户界面不友好

商业工具有以下特点：

1、获得一个指定功能和性能标准的工具的费用可能比自行开发一个工具的成本 要低

2、在文档、帮助和培训方面必须提供良好的支持

3、工具通常应该很有吸引力

4、即使使用一个商业工具，可能无法完全避免建立自己的工具

但即使决定自行开发测试工具，也不要试图生产一个可以广泛使用的商业工具。

自动化脚本之线性脚本

线性脚本是录制手工执行的测试用例得到的脚本。这种脚本包含所有用户的键盘和鼠标输入。如果仅使用线性脚本技术，每个测试用例可以通过脚本完整地被回放。线性脚本中也可能包括比较，比如检查某个窗口是否弹出。

手工运行 10 分钟的测试用例，可能需要 20 分钟到 2 个小时才能完成测试自动化的工作。因为录制的脚本需要维护，尤其是增加部分内容后的维护和测试需要花费很多时间。而且自动化以后的测试执行的时间会大于 10 分钟。

线性脚本有以下的优点：

1、不需要深入的工作或计划

2、可以加快开始自动化

3、对实际执行操作可以审计跟踪

4、用户不必是编程人员

5、提供良好的（软件或工具）的演示

线性脚本适用于以下情况：

1、演示或培训

2、执行量较少，且环境变化小的测试

3、数据转换，如将数据从 Notes 数据库中转换到 EXCEL 表格中

线性脚本有以下缺点：

1、过程繁琐
2、一切依赖于每次捕获的内容
3、测试输入和比较是“捆绑”在脚本中的
4、无共享或重用脚本
5、线性脚本容易受软件变化的影响
6、线性脚本修改代价大，维护成本高
7、非常容易受意外事件的影响，引起整个测试失败

自动化脚本之结构化脚本

结构化脚本类似于结构化程序设计，含有控制脚本执行的指令。这些指令或为控制结构，或为调用结构。控制结构中包括“顺序”、“循环”和“分支”，和结构化程序设计中的概念相同。调用结构是在一个脚本中调用另外脚本，当子脚本执行完成后再继续运行父脚本。

结构化脚本的优点是健壮性好。也可以通过循环和调用减少工作量。

结构化脚本的缺点是脚本更复杂，而且测试数据仍然“捆绑”在脚本中。

结构化脚本侧重于描述脚本中控制流程的结构化特性。

自动化脚本之共享脚本

共享脚本是指脚本可以被多个测试用例使用，一个脚本可以被另外一个脚本调用。这样可以节省生成脚本的时间；当重复任务发生变化时，只需修改一个脚本。

建立共享脚本的时间可能更长，因为需要建立更多的脚本，且每个脚本需要进行适当的修改，达到脚本共享的目的。

共享脚本可以是在不同主机、不同系统之间共享脚本，也可以是在同一主机、同一系统之间共享脚本。

共享脚本的优点有：
1、以较少的开销实现类似的测试
2、维护开销低于线性脚本
3、删除明显的重复
4、可以在脚本中增加更智能的功能

共享脚本的缺点有：
1、需要跟踪更多的脚本，给配置管理带来一定的困难
2、对于每个测试，仍然需要特定的测试脚本，因此维护费用比

较高

3、共享脚本通常是针对被测软件的某部分，存在部分脚本不能直接运行

要获得高质量的共享脚本，需要接受一定的训练。在开始编写脚本时，多花些时间进行设计是值得的。通过共享脚本技术，还可以建立脚本库，达到最大程度的共享。由于共享脚本需要被多次使用，所以与脚本相配套的文档更应该引起注意。

共享脚本侧重描述脚本中共享的特性。

自动化脚本之数据驱动脚本

数据驱动脚本技术将测试输入存储在独立的数据文件中，而不是绑定在脚本中。执行时是从数据文件而不是从脚本中读入数据。这种方法最大的好处是可以用同一个脚本允许不同的测试。对数据进行修改，也不必修改执行的脚本。

使用数据驱动脚本，可以以较小的开销实现较多的测试用例，这可以通过为一个测试脚本指定不同的测试数据文件达到。将数据文件单独列出，选择合适的数据格式和形式，可将用户的注意力集中到数据的维护和测试上。达到简化数据，减少出错的概率的目的。

数据驱动脚本的优点有：

1、可以快速增加类似的测试

2、测试者增加新测试不必掌握工具脚本语言的技术

3、对第二个及以后类似的测试无额外的维护开销

数据驱动脚本的缺点有：

1、初始建立的开销较大

2、需要专业（编程）支持

3、必须易于管理

自动化脚本之关键字驱动脚本

关键字驱动实际上是比较复杂的数据驱动技术的逻辑扩展。将数据文件变成测试用例的描述，用一系列关键字指定要执行的任务。在关键字驱动技术中，假设测试者具有某些被测系统的知识，所以不必告诉测试者如何进行详细的动作，只是说明测试用例做什么，而不是如何做。这样在脚本中使用的是说明性方法和描述性方法。描述性方法将被测软件的知识建立在测试自动化环境中，这种知识包含在支持脚本中。

例如，为完成在网页浏览时输入网址，一般的脚本需要说明在某某窗口的某某控件中输入什么字符；而在关键字驱动脚本中，可以直接是在地址栏中输入网址什么什么；甚至更简单，仅说明输入网址什么什么。

关键字驱动脚本的数量不随测试用例的数量变化，而仅随软件规模而增加。这种脚本还可以实现跨平台的用例共享，只需要更改支持脚本即可。

脚本预处理

预处理是一种或多种预编译功能，包括美化器、静态分析和一般替换。脚本的预处理是指脚本在被工具执行前必须进行编译。预处理功能通常需要工具支持，在脚本执行前自动处理。

美化器是一种对脚本格式进行检查的工具，必要时将脚本转换成符合编程规范的要求。可以让脚本编写者更专注于技术性的工作。

静态分析对脚本或表格执行更重要的检查功能，检查脚本中出现的和可能出现的缺陷。测试工具通常可以发现一些如拼写错误或不完整指令等脚本缺陷，这些功能非常有效。静态分析可以检查所有的缺陷和不当之处。类似于程序设计中的 PC-Lint 和 LogiScope 的功能。

一般替换也就是宏替换。可以让脚本更明确，易于维护。使用替换时应注意不要执行不必要的替换。在进行调试时，应该注意缺陷可能是存在被替换的部分中，而不是原来的脚本中。

自动比较技术

测试验证是检验软件是否产生了正确输出的过程，是通过在测试的实际输出与预期输出（例如，当软件正确执行时的输出）之间完成一次或多次比较来实现的。进行测试自动化工作时，自动比较就成为一个必须的环节。有计划的进行比较会比随意的比较有更高的效率和发现问题的能力。

自动比较的内容可以是多方面的，包括基于磁盘输出的比较，如对数据文件的比较；基于界面输出的比较，如对显示位图的比较；基于多媒体输出的比较，如对声音的比较；还包括其它输出的内容的比较。

比较器可以检测两组数据是否相同，功能较齐全的比较器还可以标识有差异的内容。但比较器并不能告诉用户测试是否通过或失败，需要用户自行判断。

比较可以是简单的比较，仅匹配实际输出与预期输出是否完全相同，这是自动化比较的基础。智能比较是允许用已知的差异来比较实际输出和预期输出。比如，要求比较包含日期信息的输出报表的内容。如果使用简单比较，显然是不行的，因为每次生成报表的日期信息肯定是不同的。这时就需要智能比较，忽略日期的差别，比较其它内容，甚至还可以忽略日期的具体内容，比较日期的格式，要求日期按特定格式输出。智能比较需要使用到较为复杂的比较手段，包括正则表达式的搜索技术、屏蔽的搜索技术等。