

The background features a light cream color with several large, stylized, overlapping swirls in shades of light green, light blue, and light purple. Interspersed among these swirls are numerous small, yellow, four-pointed starburst or spark-like shapes, creating a festive and dynamic feel.

六、自动化测试



内容要点

1. 自动化测试的意义
2. 自动化测试的常用技术
3. 自动化测试生存周期



1. 自动化测试的意义

- 软件测试工作量很大，其中许多操作是重复性的、非智力性的和非创造性的，并要求做准确细致的工作，计算机就最适合于代替人工去完成这样的任务。


自动化测试：

编写软件去测试其它软件。

编写驱动被测应用程序的测试脚本以执行键盘、鼠标动作和后台进程并验证应用程序响应和行为。



手工测试的局限性

- 无法做到覆盖所有代码路径
 - 机械、重复，工作量大。如果有大量（几千）的测试用例，需要在短时间内（1天）完成，手工测试几乎不可能做到
 - 许多与时序、死锁、资源冲突、多线程等有关的错误，通过手工测试很难捕捉到
 - 进行负载、性能测试，很难通过手工测试模拟大量数据或大量并发用户
 - 可靠性测试时，常需要模拟系统运行10年、几十年，以验证稳定性，这也是手工测试无法做到的
- 



自动化测试的好处

- 提高测试效率
 - 对已经建立自动化测试系统的被测软件，可以缩短软件开发测试周期，让产品更快投放市场
 - 节省人力资源，充分利用硬件，降低测试成本。
- 提高测试可重复性
 - 可运行更多，更频繁
- 改善测试质量
 - 增强测试的稳定性和可靠性
 - 提高测试的准确度和精确度，增加软件信任度
- 实现一些手工情况难以开展的测试
 - 能做负载、压力测试等手工不能做的事情



自动化测试的局限

- 测试自动化初始代价高，技术要求也高
 - 考虑开发自动化测试工具、脚本的代价，自动测试不一定减轻工作量，测试进度可能不一定缩短
- 测试工具本身缺乏想象力和灵活性，能否发现缺陷关键在于测试设计，而不在于是否自动化
- 测试工具不一定好用，一种工具难以适用于所有测试
- 自动化测试很难普遍应用，一些软件很难自动化测试，可测性低的产品也不易自动化测试
- 即使应用自动化测试，覆盖率也不会达到100%




建立正确的自动化测试目标

- 自动化测试不可能完全代替手工测试。它只是测试工作的一部分，是对手工测试的一种补充
- 自动化测试更多的是做回归测试，不能发现更多新问题，但可以验证老问题是否还存在。
- **自动化测试无智慧，有代价。**多数情况下，手工测试和自动化测试应该相结合，以最有效的方法来完成测试任务。



适合自动化测试的场合

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



单元测试、集成测试、性能测试、稳定性测试、可靠性测试等较多采用自动化测试方法；用户界面功能测试也较多自动化



不适合自动化的测试场合

➤ 那种功能不稳定的软件、开发周期很短的软件、一次性的软件等不适合开发测试工具进行自动化测试。

测试自动化建立和维护等方面的负担可能会造成延误工期、成本浪费等问题



构建高效的自动化测试体系

- 应选择合理的点进行自动化，建立合理的自动化体系结构，编写合理的自动化测试代码
- 优秀自动化测试体系的七个属性
 - 应根据测试目标，有针对性地提高自己关注的属性，实施符合需要的自动化测试体系

优秀自动化测试体系的七个属性

- **可维护性：**是否很容易使测试更新跟上软件升级的步伐
 - 软件变更很容易导致自动化脚本失效，需要建立易维护的自动化测试体系结构。
- **高效性：**自动化测试的一个重要目标就是更经济地运行测试
 - 应尽可能节省测试资源，降低对人工的依赖，等等。
- **可靠性：**测试机制是否能给出精确而且可重现的结果
- **兼容性：**是否允许测试用例为不同的测试目标而以不同方式组合
- **可用性：**是否容易使用，避免繁琐的安装配置等
- **健壮性：**是否可以处理意外情况而不退出或终止
- **可移植性：**在不同环境中运行测试的能力



2. 自动化测试常用技术

- 测试自动化实现的基础
 - 通过设计的特殊程序模拟测试人员对计算机的操作
 - 或者类似于编译系统那样对计算机程序进行检查。
- 实现测试自动化的常用技术
 - 代码静态和动态分析(Static and Dynamic Analysis)
 - 捕获和回放(Record and Replay)
 - 脚本和虚拟用户技术
 - 测试管理技术



2.1 代码静态和动态分析—白盒

- 代码分析类似于高级编译系统，一般针对不同的程序语言去构造分析工具
 - 定义类、对象、函数、变量等的使用规则，推理检查规则满足情况
 - 在分析时对代码进行语法扫描，找出不符合编码规范的地方
 - 根据某种质量模型评价代码质量
 - 基于控制流图和调用图分析时序性缺陷等



经常基于语法树(abstract syntax tree)、控制流图(control flow graph)、调用图(call graph)等来实施分析

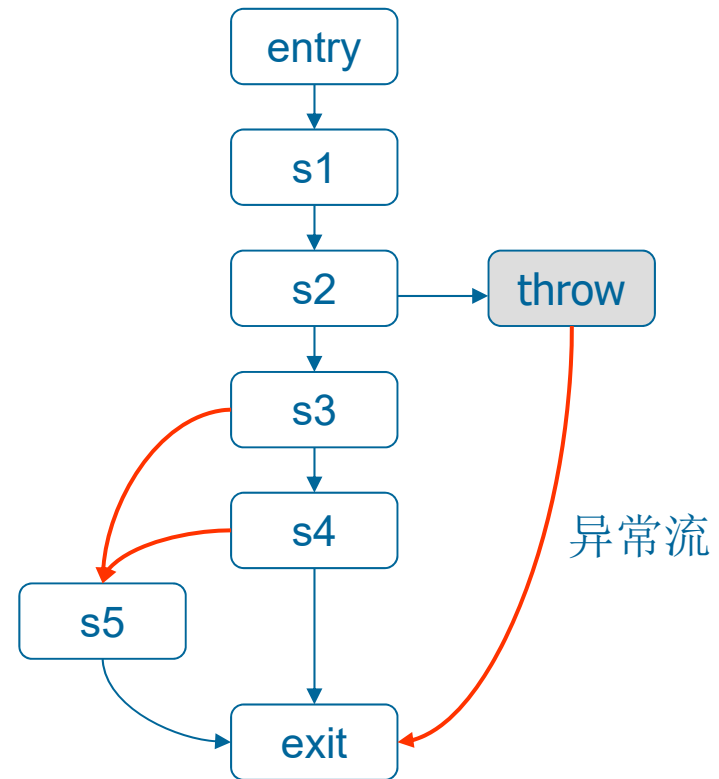


控制流图

```
void m() throws Exception{  
    s1;  
    if(s2)  
        throw new Exception();  
}
```

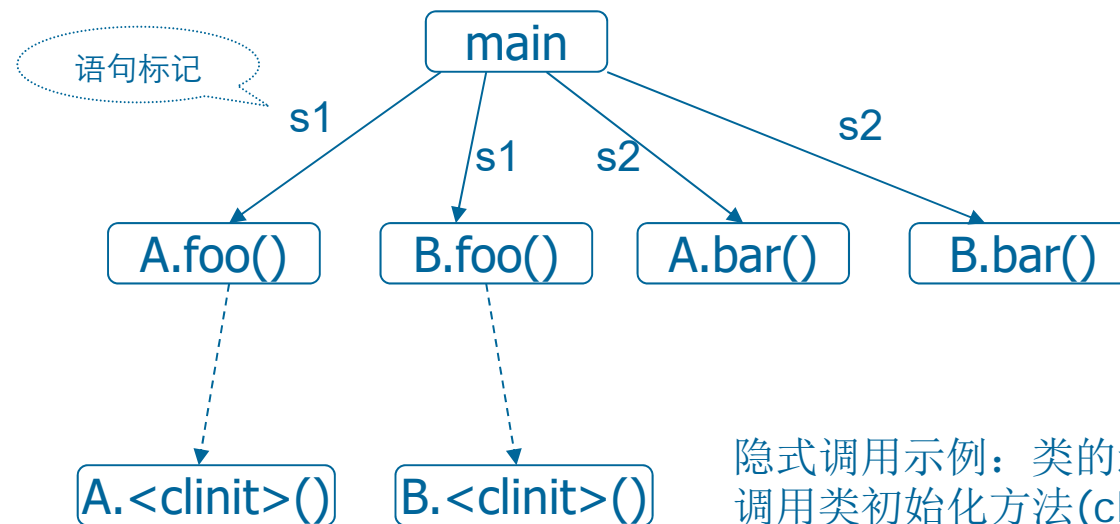
```
try{  
    s3;  
    s4;  
}  
catch(Exception e){  
    s5;  
}  
}
```

控制流图反映程序语句之间的控制转移关系，包括有异常时的情况等

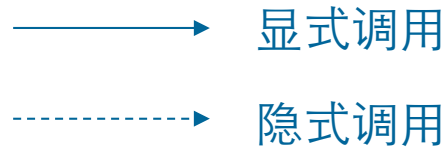


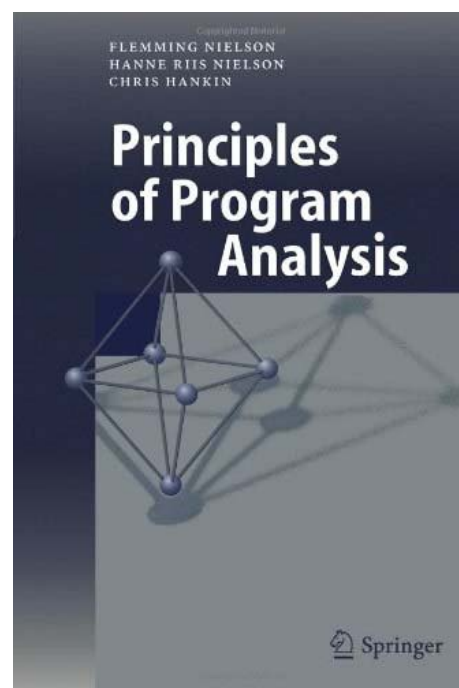
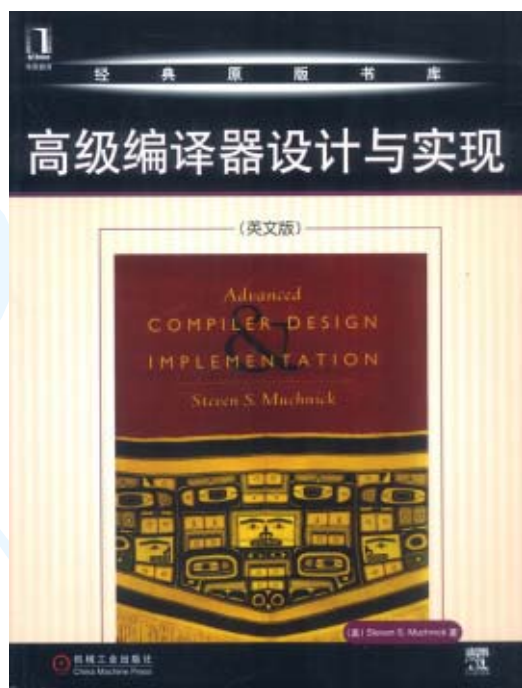
调用图

- 反映函数/方法之间的调用关系



隐式调用示例：类的运行时加载。自动调用类初始化方法(class initialize)，无调用语句





2.2捕获（录制）和回放——黑盒

捕获和回放：先由手工完成一遍需要测试的流程，同时由计算机以脚本等形式记录下这个流程，运行脚本，从而可以自动重复这个被测流程，实现自动化测试

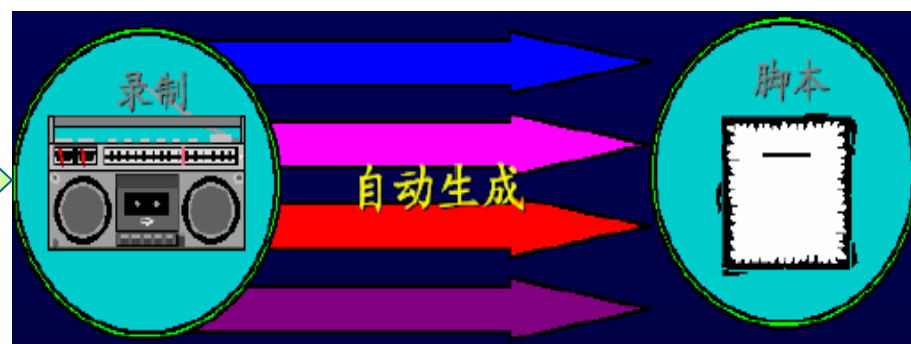
自动化测试的两种方式

手工编写测试驱动



优点：灵活性好
缺点：复杂、工作量大

录制回放方式



优点：简单、快捷
缺点：灵活性差



捕获（录制）和回放——黑盒

- 捕获

将用户每一步操作都记录下来。对GUI，记录下操作对象，以及相对应的操作、状态变化或是属性变化。所有的记录转换为一种脚本语言所描述的过程，以模拟用户操作。

操作对象描述方式主要有两种：

- 以用户界面的像素坐标描述
- 以逻辑对象（如窗口、按钮等）描述
- 以图标描述

- 回放

将脚本语言所描述的过程转换为屏幕上的操作，然后将被测系统的输出记录下来同预先给定的标准结果比较。

可大大减轻黑盒测试的工作量，能够很好地进行回归测试。

以用户界面的像素坐标描述: Android Monkey脚本

```
type= point
count= 4
speed= 1.0
start data >>
LaunchActivity(com.json.test,com.json.test.MainActivity)

captureDispatchPointer(0,0,0,524,390,0,0,0,0,0,0,0);
captureDispatchPointer(1,1,1,524,390,0,0,0,0,0,0,0);
UserWait(300);
captureDispatchPointer(0,0,0,231,648,0,0,0,0,0,0,0);
captureDispatchPointer(1,1,1,231,648,0,0,0,0,0,0,0);
UserWait(300);
captureDispatchPointer(0,0,0,698,231,0,0,0,0,0,0,0);
captureDispatchPointer(1,1,1,698,231,0,0,0,0,0,0,0);
UserWait(300);
captureDispatchPointer(0,0,0,100,200,0,0,0,0,0,0,0);
captureDispatchPointer(1,1,1,100,200,0,0,0,0,0,0,0);

# Shell_TrayWnd
set_window ("Shell_TrayWnd", 0);
button_press ("开始");

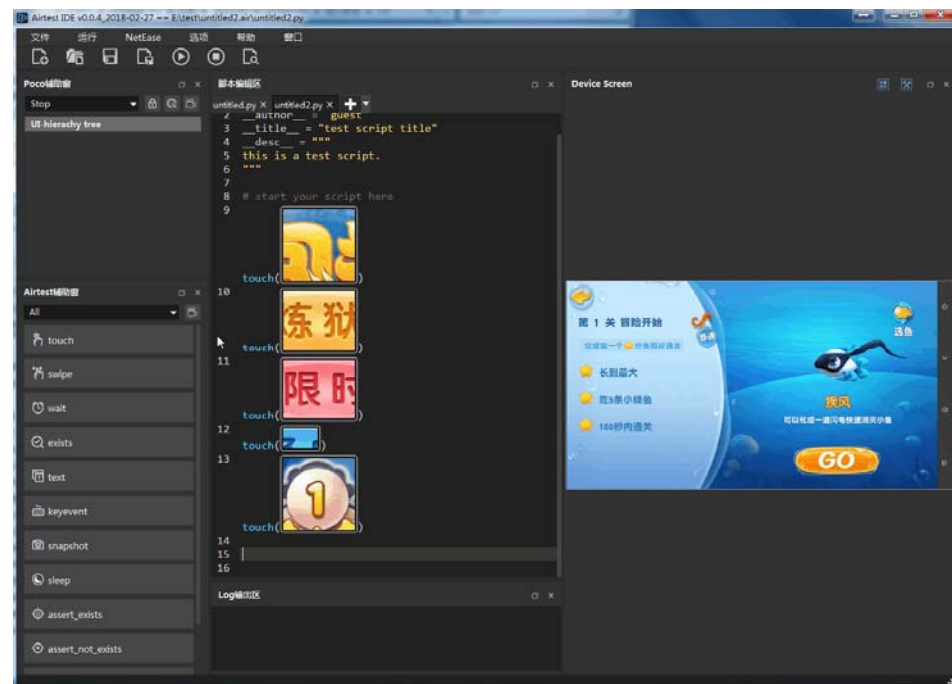
# BaseBar
set_window ("BaseBar", 6);
toolbar_select_item ("ToolbarWindow32_1", "程序(P);附件;计算器");

# 计算器
win_move ("计算器", 450, 295);
set_window ("计算器", 7);
button_press ("7");
button_press ("+");
button_press ("8");
button_press ("=");
win_close ("计算器");
```

以逻辑对象描述: WinRunner脚本



sikuli





Airtest

以图标描述



2.3. 脚本技术

- 脚本是一组测试工具执行的指令集合，也是计算机程序的一种形式。
 - 主要用于测试Web、GUI、通信等
 - 脚本可以通过录制测试产生，然后再做修改（减少直接编写的工作量）；也可以直接用脚本语言编写
 - 脚本语言：Python, vbscript, javascript, C子集等
- 
- 

脚本技术

- 脚本技术可分为以下几类:

- 线性脚本

- 结构化脚本

- 共享脚本

- 数据驱动脚本

- 关键字驱动脚本



脚本技术

- 脚本技术可分为以下几类：

- **线性脚本：** 录制手工执行的测试用例得到的脚本

- 缺点： **结构为顺序语句**，不具有逻辑判断能力，可维护性差，代码量大，兼容性差

- 优点：简单，易得。用作软件演示很不错

脚本示例：类C语言脚本

```
# Shell_TrayWna
    set_window ("Shell_TrayWnd", 0);
    button_press ("开始");

# BaseBar
    set_window ("BaseBar", 6);
    toolbar_select_item ("ToolbarWindow32_1", "程序(P);附件;计算器");

# 计算器
    win_move ("计算器", 450, 295);
    set_window ("计算器", 7);
    button_press ("7");
    button_press ("+");
    button_press ("8");
    button_press ("=");
    win_close ("计算器");
```



1. 点击”开始”菜单
2. 选择菜单目录
“程序：附件：计算器”
3. 将计算器移动到屏幕
中间位置
4. 按键计算 $7+8=15$
5. 关闭计算器



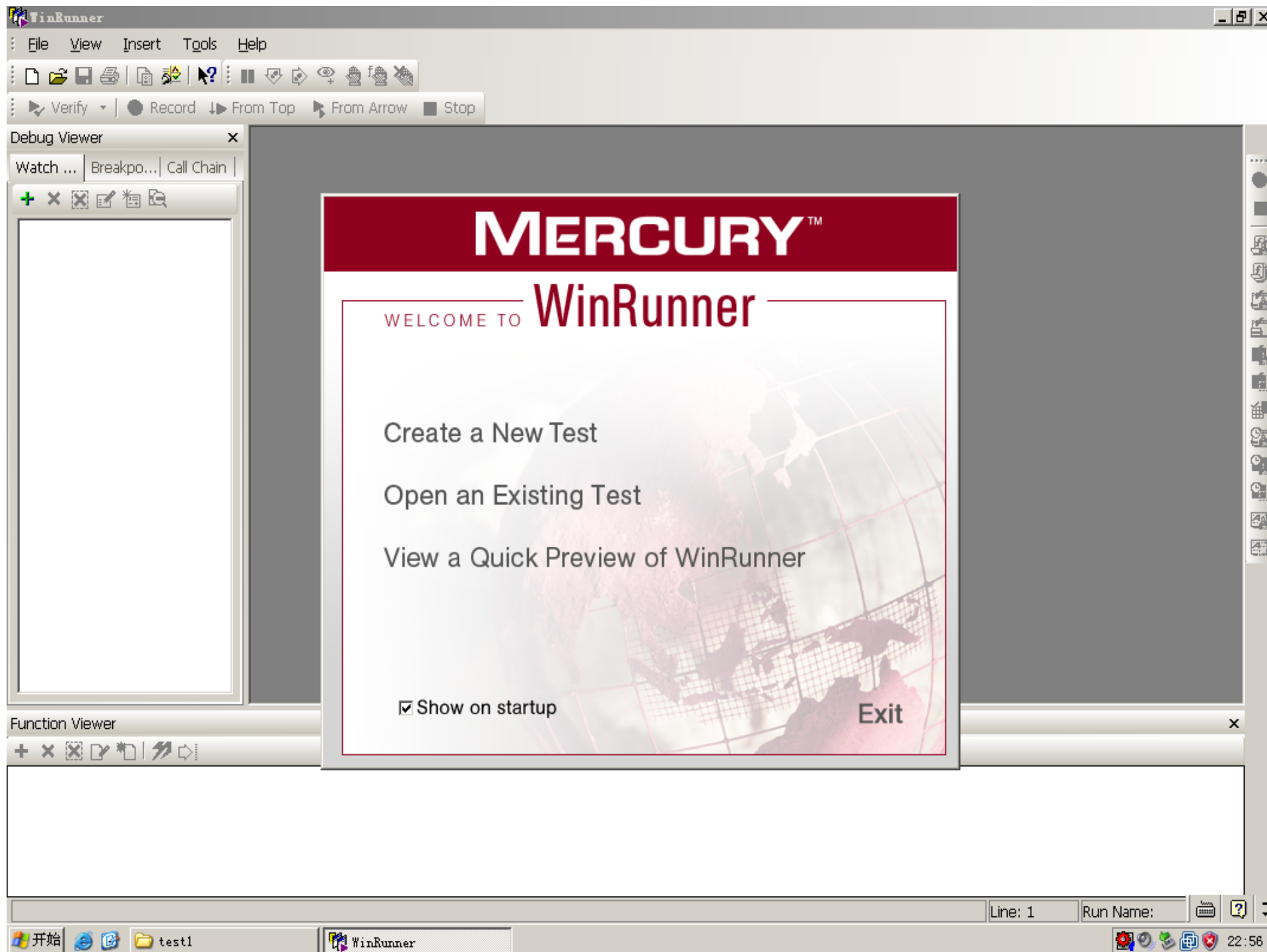
基于脚本的GUI测试示例

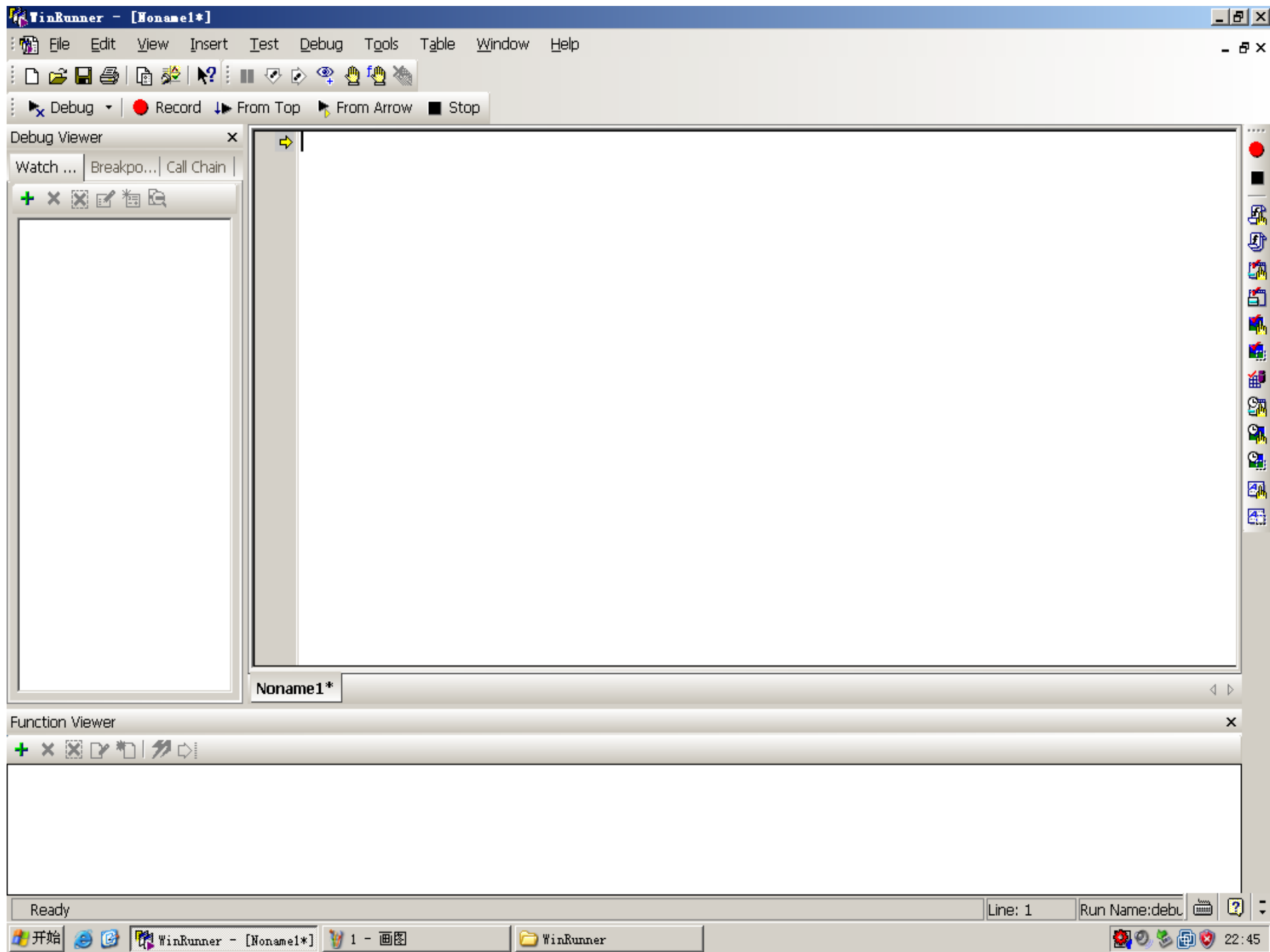
- 典型工具

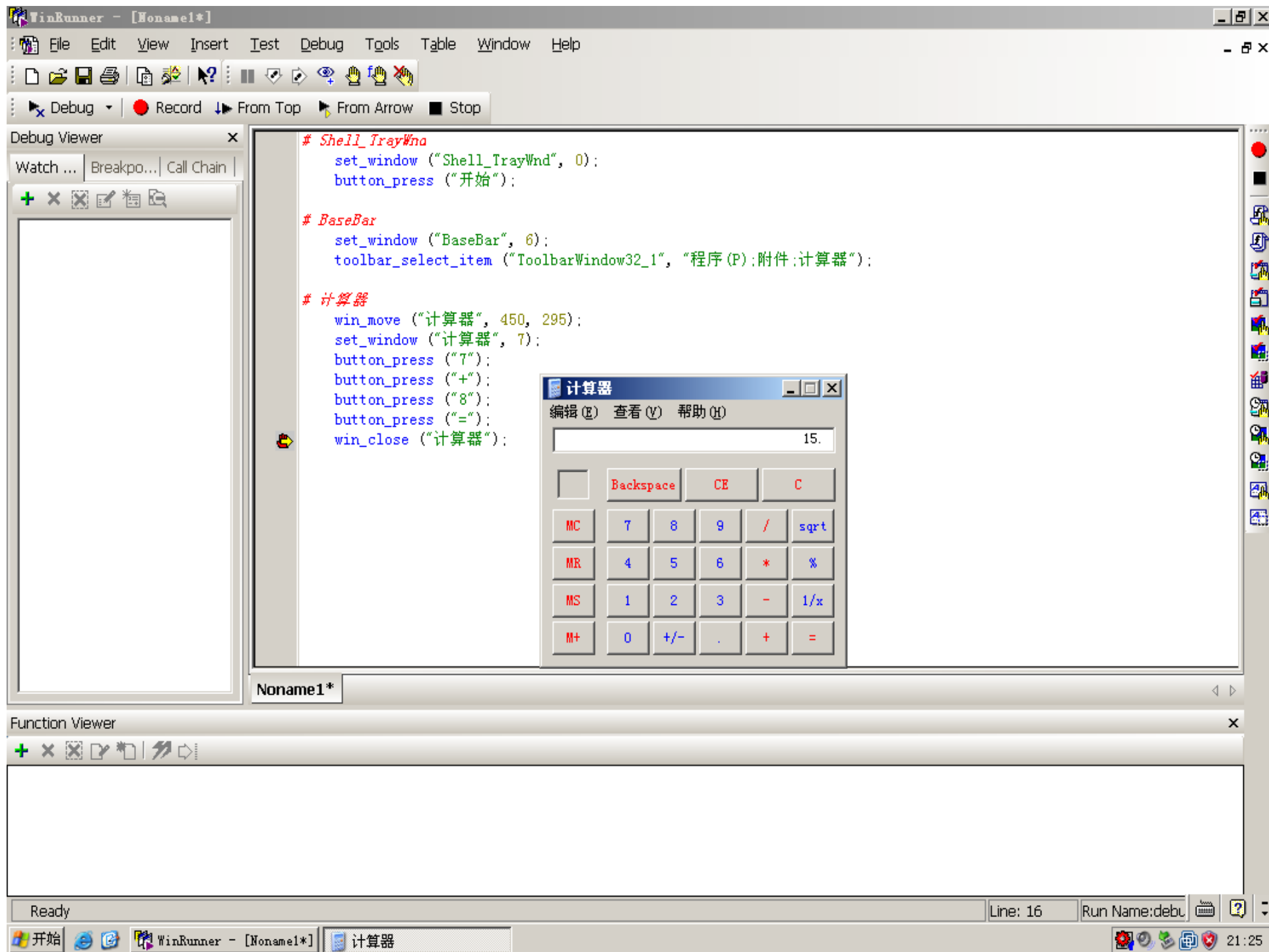
- Windows: Quick Test Pro(QTP), Rational Functional Tester (RFT), WinRunner
- Android: UiAutomator、Robotium、Appium
- Web: Selenium

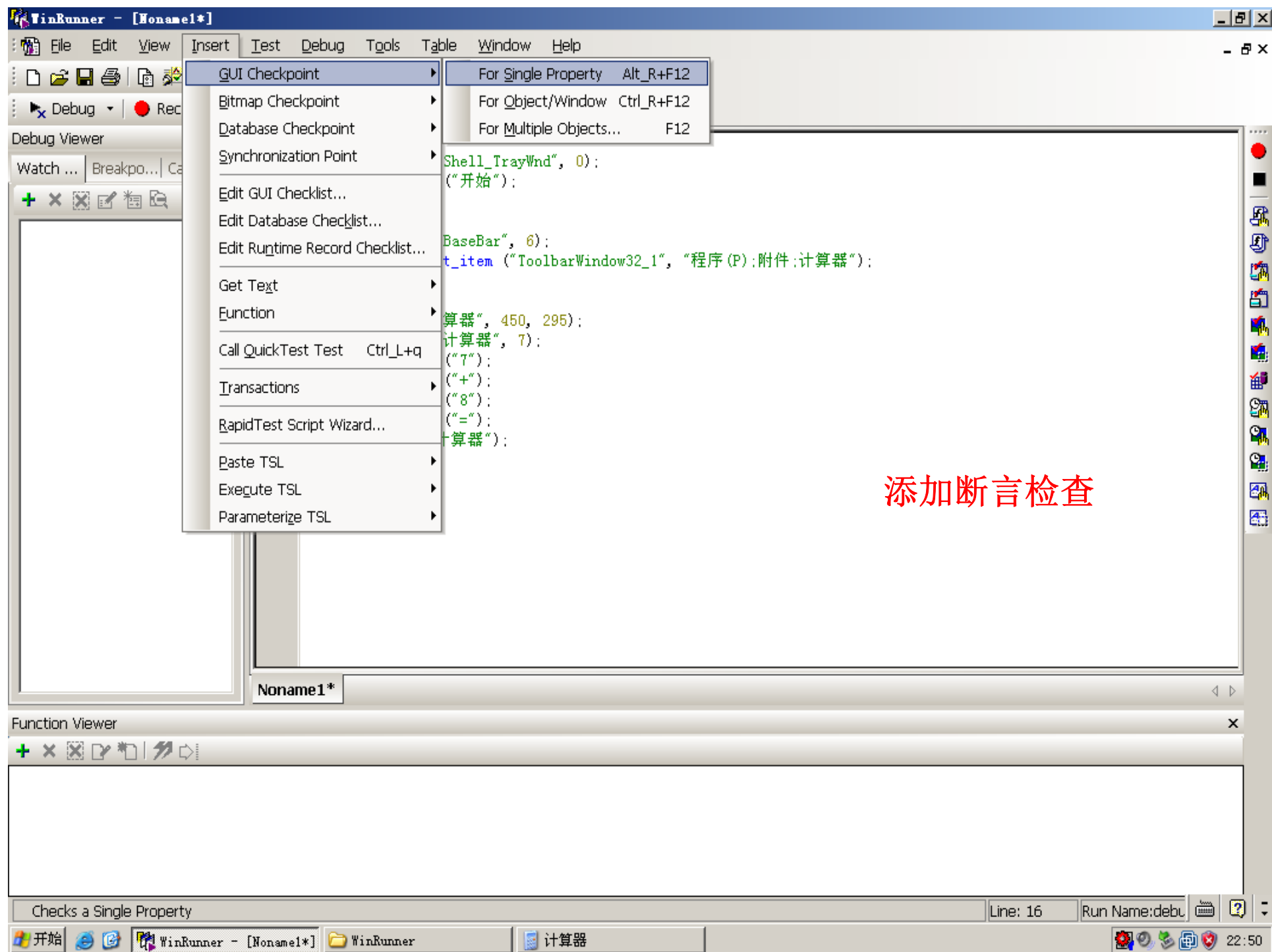
- WinRunner

- 一个企业级的功能测试工具。以类C语言测试脚本形式将业务的过程（主要是GUI操作）记录下来，用以自动化执行，实施自动化测试。









Check Single Property

Click on the selected object/window.
Click the right mouse button to quit.



我的电脑



网上邻居



Internet
Explorer



回收站



lservrc



WinRunner



开始



WinRunner - [Noname1*]



WinRunner



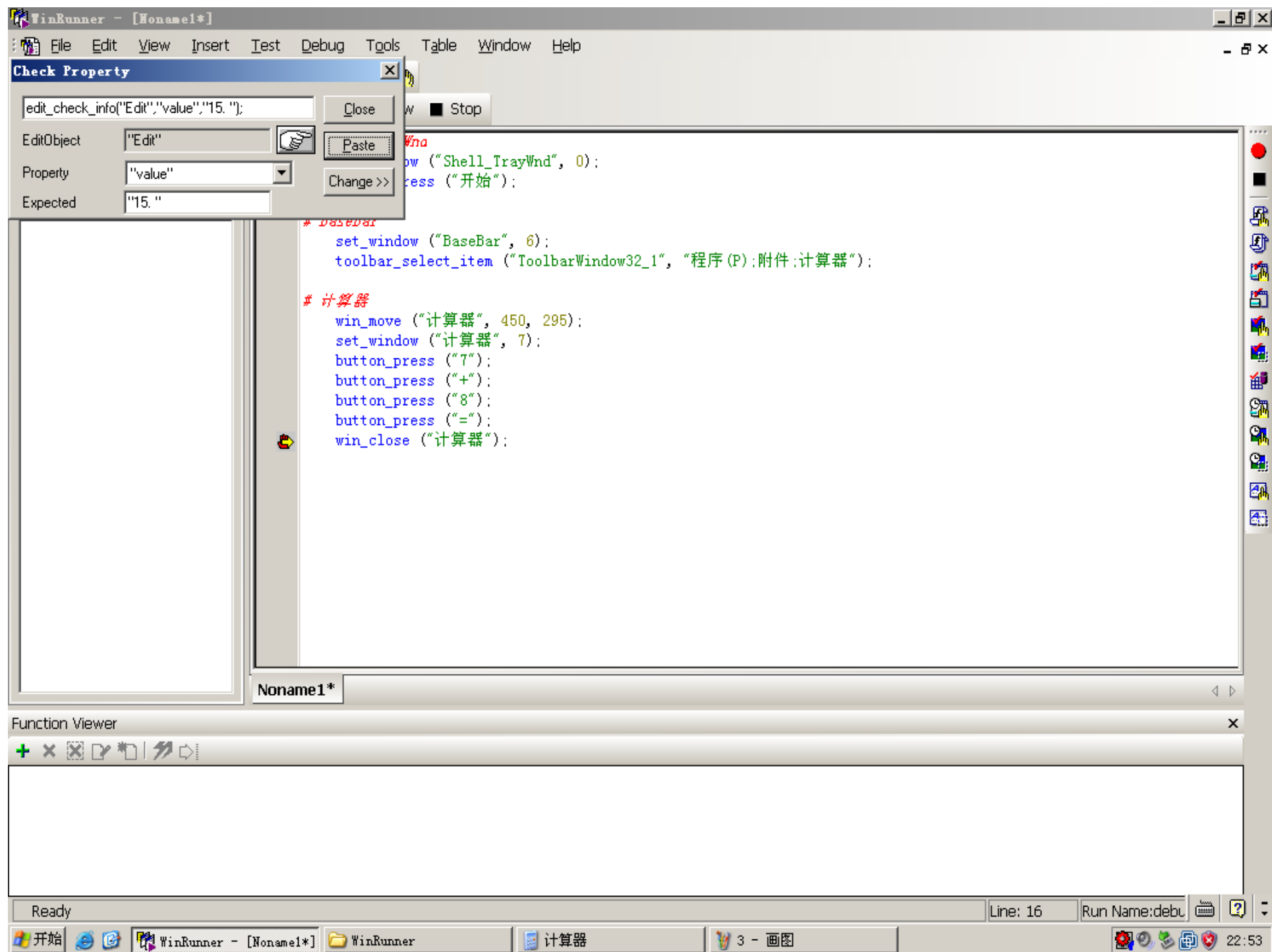
计算器

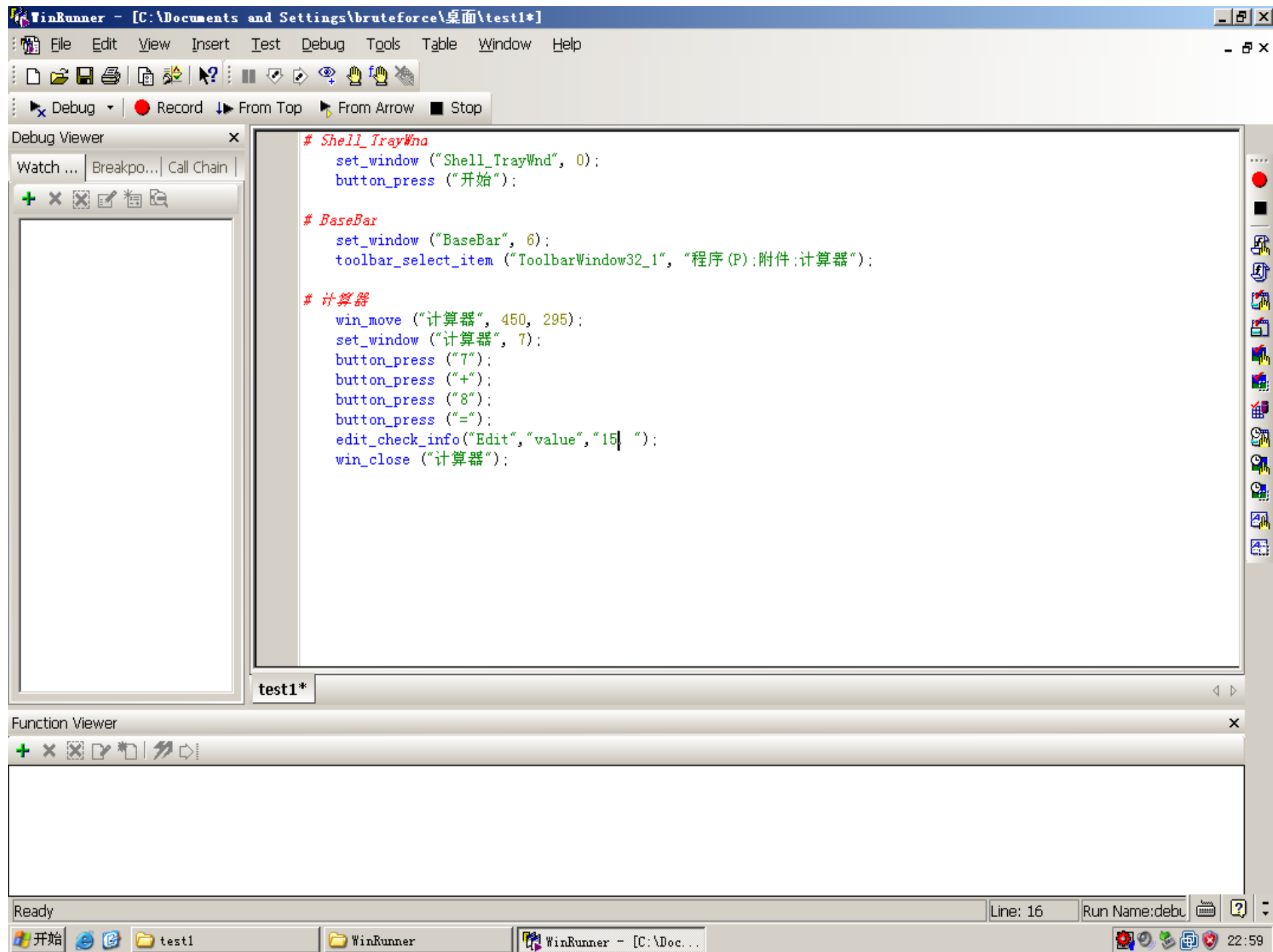


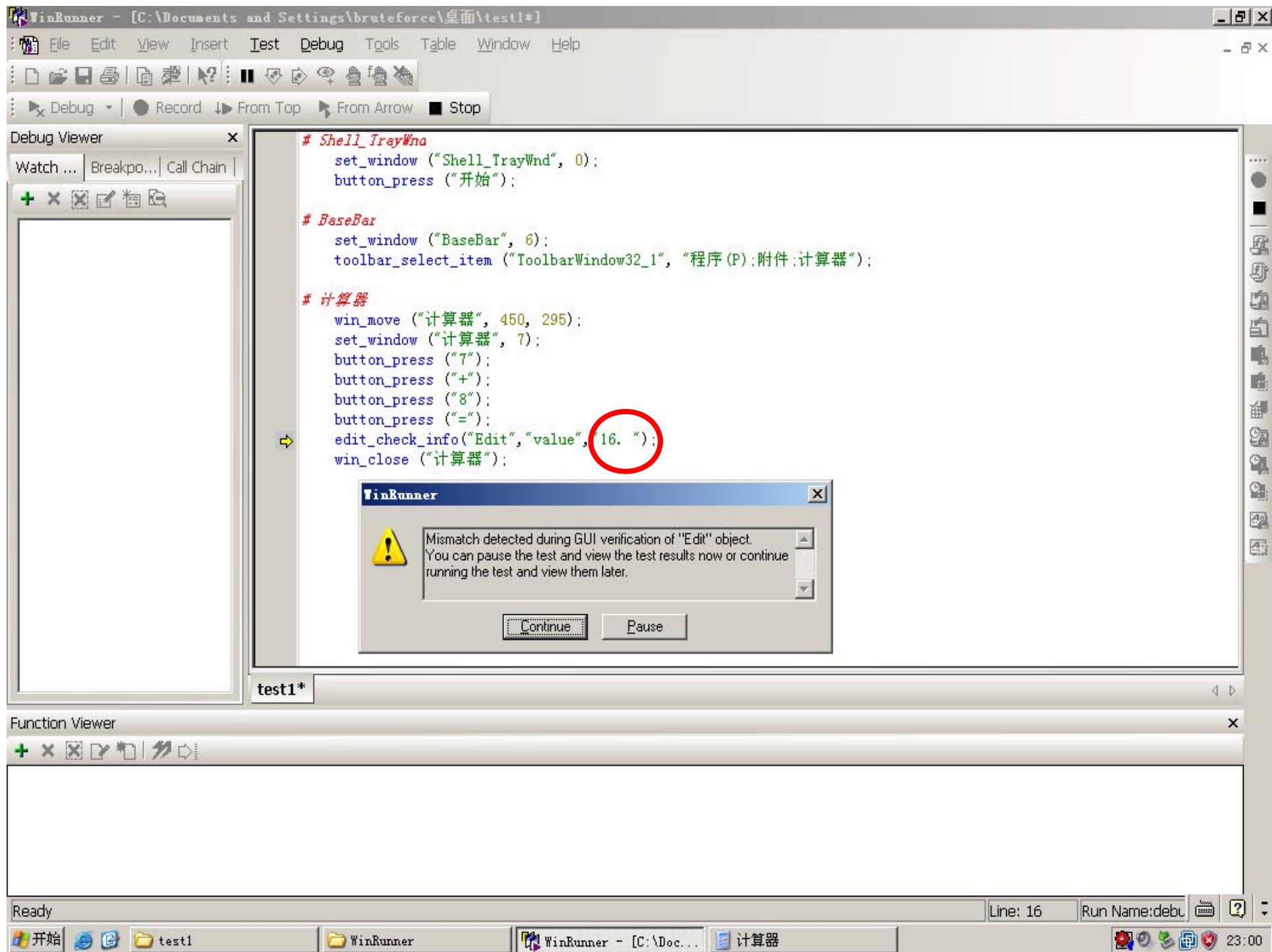
2 - 画图



22:53









脚本技术

- 脚本技术可分为以下几类：

- **结构化脚本：**类似于结构化程序，具有各种逻辑结构（顺序、分支、循环），甚至函数调用功能

- 优点：脚本的逻辑判断能力和处理问题的灵活性得到增强，基本实现脚本的模块化。

- 缺点：脚本内仍然捆绑测试信息，即键盘、鼠标动作表示的输入被固化在脚本中，测试修改和定制非常复杂困难。

脚本示例：Python脚本（实验14）

```
import time
from splinter import Browser
```

splinter是Selenium的一个封装

```
#打开浏览器，默认为 firefox
with Browser() as browser:
```

```
    # Visit URL
```

```
    url = "http://www.baidu.com"
```

```
    browser.visit(url)
```

```
    #wd 代表百度首页的搜索的输入框，可以通过右击输入框选择检查来查看
```

```
    browser.fill('wd', 'splinter - python acceptance testing for web applications')
```

```
    # 点击'百度一下'这个按钮
```

```
    button = browser.find_by_id('su')
```

```
    button.click()
```

```
    # 防止浏览器响应很慢，增加一个暂停
```

```
    time.sleep(10)
```

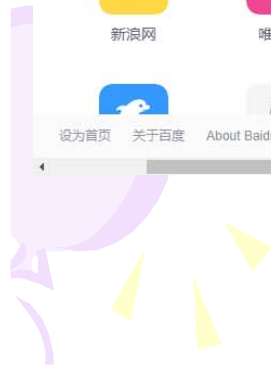
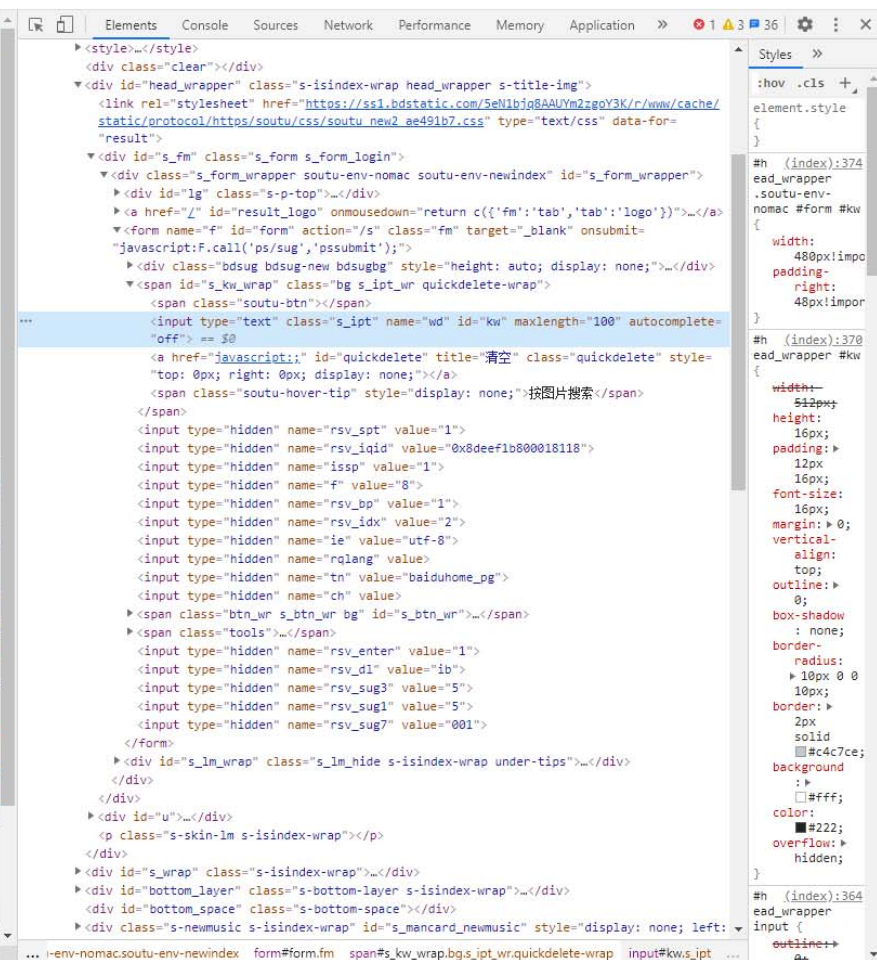
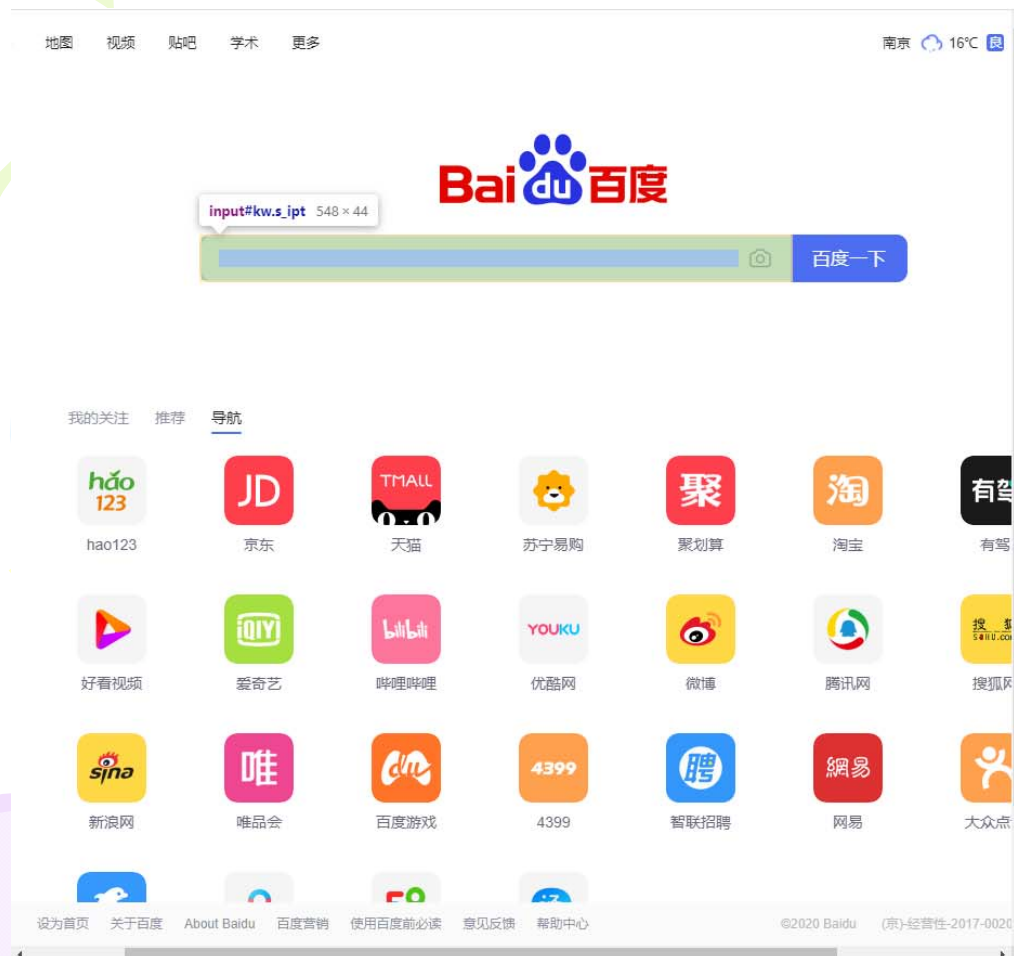
```
    #检查浏览器 url 是否有'splinter.readthedocs.io'，这是 Splinter 官网
```

```
    if browser.is_text_present('splinter.readthedocs.io'):
```

```
        print("Yes, the official website was found!")
```

```
    else:
```

```
        print("No, the official website wasn't found. ")
```



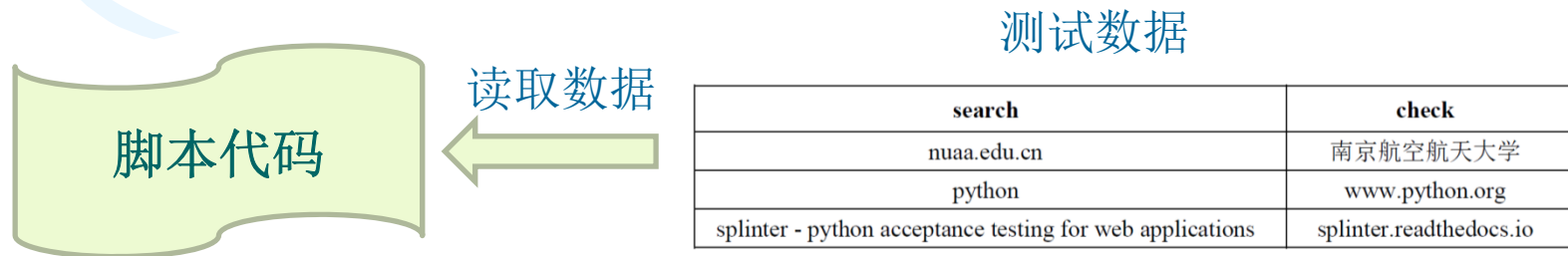
脚本技术

- 脚本技术可分为以下几类：

➤ **共享脚本：** 指某个脚本可被多个测试用例使用，即脚本语言允许一个脚本调用另一个脚本（类似C语言可以加`#include`）

➤ **优点：** 可将公共操作放在一个地方提供调用实现共享，可重用性加强，在需要更改此部分的代码时，不用修改所有使用此公共操作的脚本，大大减少了维护开销。

➤ **数据驱动脚本：** 将测试输入存储在独立的数据文件中。数据和执行控制分离（大量测试用例其操作步骤相同，使用数据不同）



测试数据结构清晰，很容易修改

脚本技术

- 脚本技术可分为以下几类：
 - 关键字驱动脚本：用关键词序列表示测试
 - 优点：测试描述与测试实现分离；脚本与数据分离；测试流程与基本测试步骤解耦，步骤可组合
 - 缺点：需要有驱动程序去解释测试描述，将关键词序列变为可执行的程序

表14-2 关键字文件Keyword.xlsx

action	Param1	Param2
open	http://baidu.com	Testing
nextPage		
selectItem	1	

动作和数据放在关键字文件中，单独编写脚本去解释每个动作

每个测试用例就是一个关键字及其参数的简表，关键字可轻松组合



关键字驱动脚本的编写 (以基于Selenium的测试为例)

- **Selenium**基本用法

```
from selenium import webdriver  
import time
```

```
browser = webdriver.Chrome()
```

```
browser.get(url='https://www.baidu.com')  
time.sleep(2)
```

```
input = browser.find_element_by_css_selector('#kw')  
input.send_keys('软件测试')
```

```
button = browser.find_element_by_css_selector('#su')  
button.click()  
time.sleep(10)
```

```
browser.close()
```

find_elements_by_name

find_elements_by_id

find_elements_by_xpath

find_elements_by_link_text

find_elements_by_partial_link_text

find_elements_by_tag_name

find_elements_by_class_name

find_elements_by_css_selector



关键字驱动脚本的编写 (以基于Selenium的测试为例)

1. 编写每个关键字的解释程序

打开给定网址并搜索

```
def openBrowser(firstParam, secondParam):  
    driver.get(url=firstParam + "/")  
    driver.find_element_by_id("kw").click()  
    driver.find_element_by_id("kw").clear()  
    driver.find_element_by_id("kw").send_keys(secondParam)  
    driver.find_element_by_id("su").click()
```

跳转到下一页

```
def goNextPage():  
    driver.find_element_by_link_text(u"下一页>").click()
```

选取搜索结果中的项

```
def select(firstParam, secondParam):  
    list = driver.find_elements_by_xpath('//div/h3/a')
```

解释关键字open、nextPage、selectItem

关键字驱动脚本的编写 (以基于Selenium的测试为例)

2. 编写总的关键字文件读取和解释框架

读取关键字文件到list

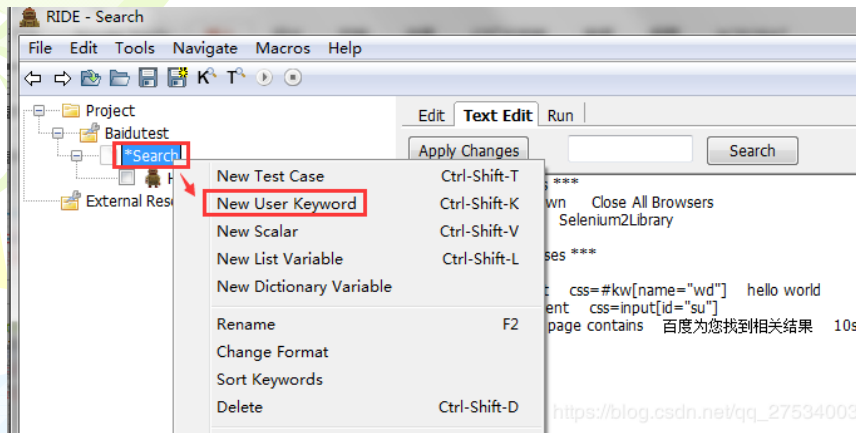
```
# 程序入口
if __name__ == '__main__':
    driver = webdriver.Firefox()
    list = read_table()
    for k in range(len(list)):
        job = list[k]
        doTestJob(job)
        time.sleep(5)

    driver.quit()
```

按关键字文件动作list进行操作

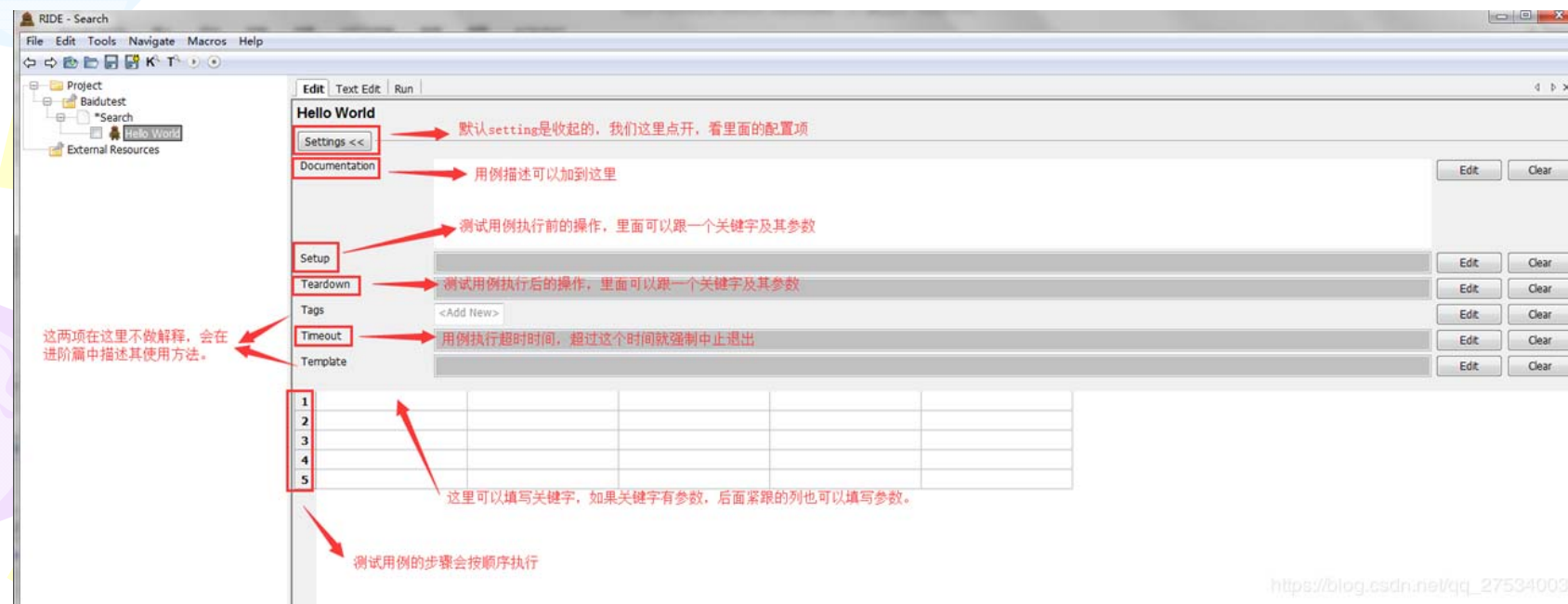
```
# 按照关键字执行测试任务
def doTestJob(job):
    jobType = job[0]
    if jobType == 'open':
        openBrowser(job[1], job[2])
    elif jobType == 'nextPage':
        goNextPage()
    elif jobType == 'selectItem':
        select(job[1], job[2])
```

Robot Framework基于关键词的测试框架



系统自带关键词
并可自定义关键词

基于关键词定义测试用例





测试脚本编写注意事项

- 对软硬件环境和测试执行的前提条件进行验证
- 稳定，能处理各种可能的意外，避免测试中断
- 能对有关环境设置进行彻底清除，避免影响后续测试用例的运行
- 各测试用例有相对独立性，不相互依赖，便于进行各种运行组合
- 有好的日志，如发生意外，可据日志了解发生的情况
- 设计简洁，代码可读性高
- 易于运行
- 具有一定的灵活性，能快速适应产品功能设计的更改

3. 自动化测试生存周期

包括制订测试计划、建立测试环境、测试设计和设计开发。

4. 测试计划、设计与开发

包括测试过程分析和测试工具的考查

3. 自动化测试引入过程

测试执行与管理

自动化测试
生存周期

测试工具获取

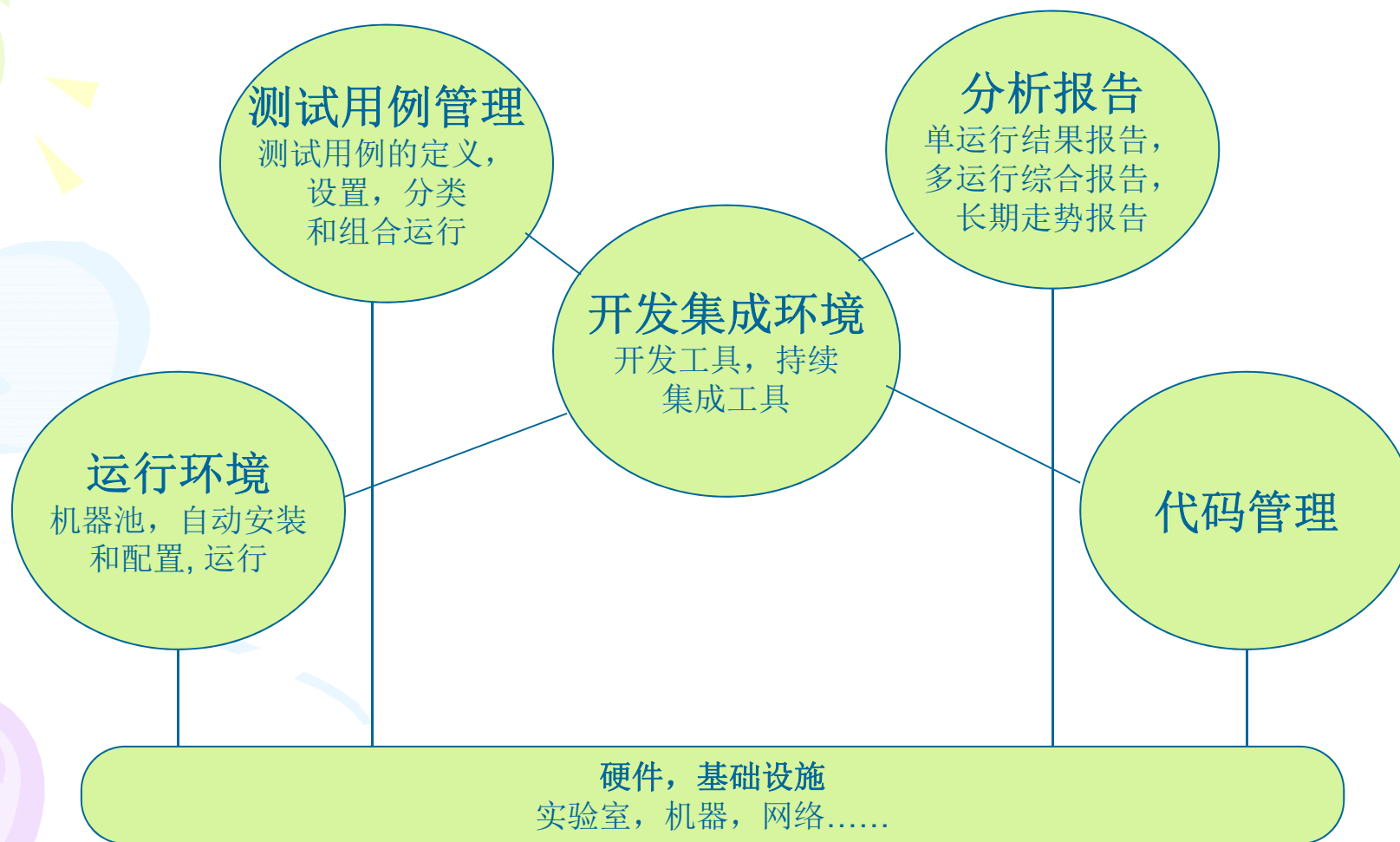
6. 测试评审与评估

1. 自动化测试决定

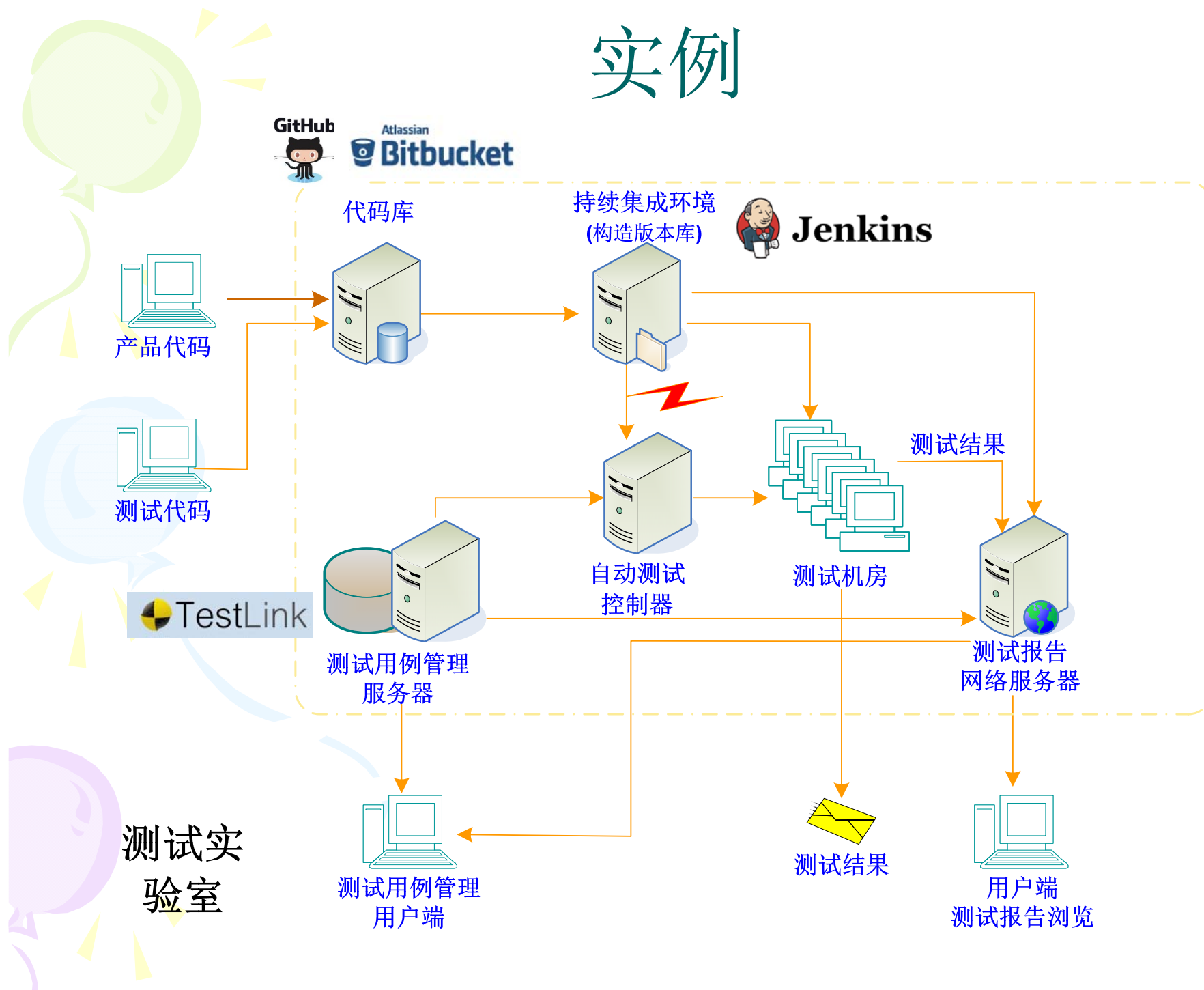
测试自动化的发展阶段与计划

	无自动测试阶段	自动测试初始阶段	自动测试发展阶段	自动测试成熟阶段	自动测试高级阶段
特征	<ul style="list-style-type: none">• 无自动测试用例，测试工作全部手工操作• 非专业人员从事测试	<ul style="list-style-type: none">• 有脚本文件驱动的半自动测试• 无机算计软件 专业人员从事测试• 无工具，无实验室	<ul style="list-style-type: none">• 有部分测试用例全部自动化• 有机算计软件 专业人员从事测试代码的开发，使用Java，C#语言• 使用工具和专门的实验室测试环境	<ul style="list-style-type: none">• 大量测试用例全部自动化• 有自己开发的共用代码库• 有测试用例自动运行系统，并与产品建造系统结合• 有测试报告和统计分析服务	<ul style="list-style-type: none">• 有高水平的测试开发人员，测试架构师• 有高度自动化的实验室系统，和专业的系统管理队伍• 有完善的自动化测试流程• 能对外提供自动测试的商业服务
计划和措施	<ul style="list-style-type: none">• 培训和引进人才	<ul style="list-style-type: none">• 培训和引进机算计软件专业人才• 尝试测试工具• 建立实验室	<ul style="list-style-type: none">• 培训和引进高级编码人才• 由开发人员帮助设计测试代码库• 系统的选择使用和整合各种工具	<ul style="list-style-type: none">• 培训和引进测试设计和架构人才• 进一步提高实验室系统• 建立流程模型和商业服务模型	

完备的自动化测试体系

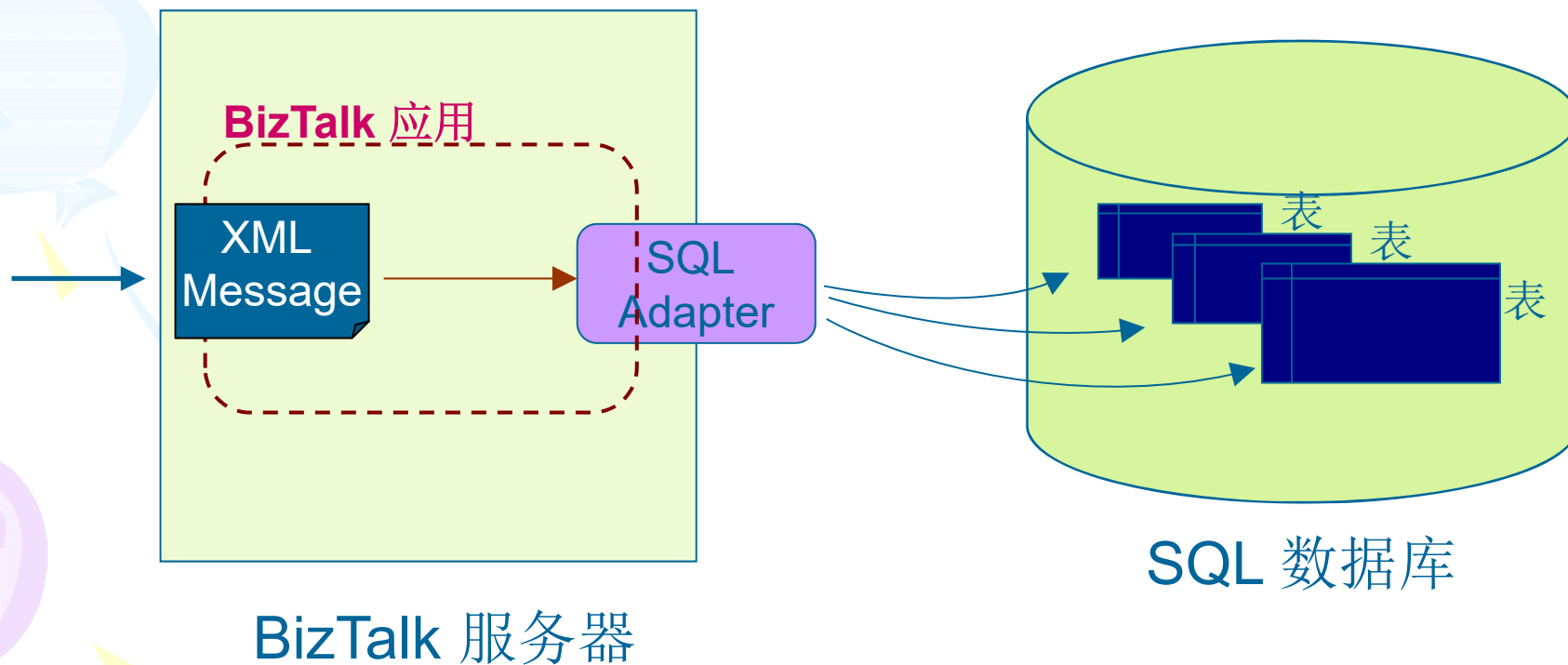


实例



每日构建+自动化测试

案例: BizTalk SQL Adapter





每日构建 + 自动化测试

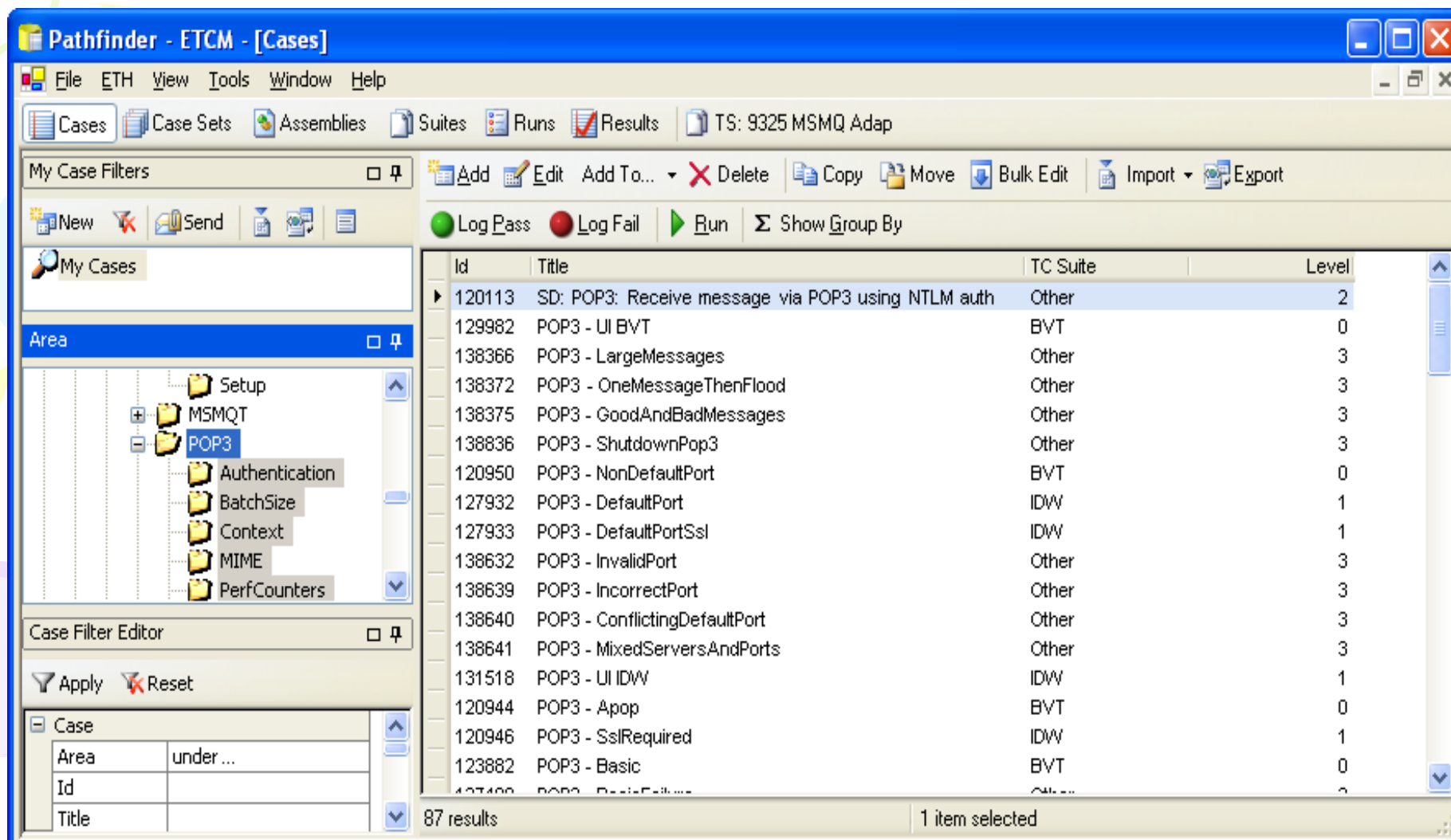
- 版本**Check Out**: cvs/svn/SourceSafe/git/...
- 项目构建: make/msdev/ant/....
- 测试环境的搭建
 - 安装并验证BizTalk、SQL数据库等已安装设置好
- 测试目标的安装与配置
 - 在BizTalk中设置一个使用SQL Adapter的应用
 - 在SQL数据库中建立测试表格
- 运行测试
 - 生成一个预定格式的Message: XML字符串, 发送给BizTalk
- 检验结果
 - 访问SQL数据库, 验证数据到达无误; 验证系统日志中无意外事件
- 清理
 - 删除BizTalk中的应用; 删除SQL中的表格
 - 其他意外情况下的特殊清理

测试用例的分级，及运行频度

	BVT	第一级	第二级	高级
定义	最简单的功能验证	所有非出错情形下的代码路径	所有出错情形下的代码路径	压力，性能，安全性.....
运行频度	每日	隔日	每周	根据需要

注：BVT 是Build Verification Test的缩写

测试用例管理



Pathfinder - ETCM - [TC: 104437 *]

File ETH View Tools Window Help

Cases Case Sets Assemblies Suites Runs Results TS: 9325 MSMQ Adap TC: 104437 *

Save and Close Save and New Cancel Previous Case Next Case Log Pass Log Fail

Title: SchemaWizard_BVT_SelectReceive

Namespace:

Path: \Voyager\Engine\Messaging Engine\SQL Adapter\BVT\

Level: 0: Acceptance test TC Suite: BVT Created By: jeffwang (6/18/2003)

Owner: jeffwang Method: Automated Updated: jeffwang (8/30/2005)

Type: UI, BizDesk, MMC Time to run: 0 minutes

Status: Active

☐ Globalization Coverage ☐ Integration ☐ Security

Automation Properties

Script Type: Command Prompt Timeout: 975 seconds

Script: cmd /c AdapterTest -TEST %TestDir%\CFG\SchemaWizard_BVT_SelectReceive.xml %TestDir%\%TestOutput%
Note: Command line MUST start with "cmd /c" for CMD and BAT files.

Description Bugs (0) Log Parameters: (0) Suites (1) Test Case Sets (0) Configurations

Log Filter Editor

Apply Reset

Result

Result	in (Fail, Check)
Tester	= jeffwang
Date	
Comment	
Resolution	= Unresolved

Test Run

Machine ...	
Build	
Configurat ...	

Open Open Run Open in Log Resolve Assign Add Comment Show Group By

Id	Build	Configuration	Tester	Result	Date	Resolution
11093...	7237.00	EN/MENU - Win...	jeffwang	FAIL	3/9/2005 ...	Unresolved
11221...	7237.00	EN/KOR - Win...	jeffwang	FAIL	3/9/2005 ...	Unresolved
11487...	7239.00	EN/MENU - Win...	jeffwang	FAIL	3/11/2005 ...	Unresolved
11820...	7245.00	EN/MENU - Win...	jeffwang	FAIL	3/15/2005 ...	Unresolved
11835...	7246.00	EN/KOR - Win...	jeffwang	FAIL	3/16/2005 ...	Unresolved
11838...	7246.00	EN/MENU - Win...	jeffwang	FAIL	3/16/2005 ...	Unresolved
11890...	7246.00	EN/MENU - Win...	jeffwang	FAIL	3/16/2005 ...	Unresolved
12044...	7233.00	EN/MENU - Win...	jeffwang	FAIL	3/16/2005 ...	Unresolved
12059...	7238.00	EN/MENU - Win...	jeffwang	FAIL	3/16/2005 ...	Unresolved

255 results 1 item selected



测试报告

Project Explorer - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Home Search Favorites

Address <http://bpiweb/Apps/ProjectExplorer/Test/Run/?p=96&r=226872> Go Links

bpiweb Business Process and Integration Division **Microsoft**

My Site Site Settings Help

Home Topics News Sites Help

BPI Web > Tools > Project Explorer > VoyagerSP1 Home > - JTP BVT Tests WSI Official

VoyagerSP1 - JTP BVT Tests WSI Official

Summary

Configuration:	EN/ENU - Win 2003 - Ent - Sql:STANDARD	Reported Pass/Fail:	771 / 73	Tasks
Build:	3.0.6070.0 retail [en]	Cases logged to ETCM:	Not logged	Log test case results to ETCM
Started:	Thursday Sep 1 - 4:09 AM	Run Type:	Private	Change to an official test run
Running Time:	5 hours 19 mins	Reporting Label:	(none set)	Change reporting label

Log File Log Viewer

Engine subsuite\ Rules Engine Official BVT\ Run Single Test Simple Orchestration (SimpleOrder.1)	0	1	230	0	leiding	...	120 sec.
Engine subsuite\ Web Services Orchestration BVT Official BVT	74	51	13055	0	narpad	...	1296 sec.
Engine subsuite\ MSMQT Official BVT	1	0	241	1	iuliur	...	95 sec.
Engine subsuite\ Transports Official BVT Suite	20	5	1206	4	karahanc	...	2176 sec.
Environment for this suite Link to First Failure for this suite(if any)							
cmd /c Cathi_CopyTest Scripts\WSI\TestCaseRuntime*.bat	0	0	15	0	valerik	valerik	1 sec.
cmd /c Cathi_CopyTest Binaries\WSI\NetisLib	0	0	21	0	valerik	valerik	1 sec.
cmd /c Cathi_CopyTest Binaries\WSI\Tests\Common	0	0	15	0	dstucki	dstucki	1 sec.
cmd /c Cathi_CopyTest Binaries\WSI\Tests\Transports	0	0	64	0	dstucki	dstucki	2 sec.

BPID Tools Microsoft Confidential Server: BPIDWEB1

Local intranet

✉ UPDATE w/ today's results : Test Pass Day 4/Week1 - Pathfinder Official Build 3.5.1067.0 - test runs are DELAY...

File Edit View Insert Format Tools Actions Help

Reply Reply to All Forward [Icons]

From: BTS Automation

Sent: Thu 6/16/2005 3:46 PM

To: BTS Automation Information

Cc:

Subject: UPDATE w/ today's results : Test Pass Day 4/Week1 - Pathfinder Official Build 3.5.1067.0 - test runs are DELAYED

BVT runs De Jour

32bit BVT - W2K3 Yukon EN
32bit BVT - XP Yukon EN
32bit BVT - W2K3 w/Binary Collation EN
64bit(WOW) BVT - W2K3 Yukon EN

BVT Test Coverage - Overall 64.23% pass, 68.4% complete

EN/ENU - Win 2003 AMD64 - BizTalk Enterprise -

Area	Pass	Fail	NR
Application Deployment	37	2	0
BMI	0	0	400
EDI	0	0	154
EndToEnd	2	0	0
Engine	277	27	36
ITPro	297	12	0
Setup-Admin-SDK	241	0	0
Tools	85	2	0
Tracking	26	5	0
Upgrade-Setup-Config	1	1	28
Total:	966	49	618

EN/ENU - Win 2003 x86 - BizTalk Enterprise - SQL 8.00.761 Latin1_General_BIN

Area	Pass	Fail	NR
Application Deployment	39	0	0
BMI	0	0	345
EDI	26	128	0
EndToEnd	0	0	2
Engine	61	1	278
ITPro	307	0	0
Setup-Admin-SDK	223	18	0
Tools	87	0	0
Tracking	31	0	0
Upgrade-Setup-Config	9	1	20
Total:	783	148	645

EN/ENU - Win 2003 x86 - BizTalk Enterprise - SQL 9.00.1187.07 Latin1_General_CI_AI

Area	Pass	Fail	NR
Application Deployment	39	0	0
BMI	339	27	34
EDI	0	0	154
EndToEnd	1	1	0
Engine	339	1	0
ITPro	301	8	0
Setup-Admin-SDK	241	0	0
Tools	85	2	0
Tracking	28	3	0
Upgrade-Setup-Config	10	1	19
Total:	1383	43	207



自动化测试工具的常见功能

- 测试用例的生成：生成测试输入、预期输出、操作指令等
- 测试的执行与控制：包括单机运行和网络多机分布式的运行，在节假日的运行，测试用例执行控制，测试对象、测试范围与测试版本的控制等
- 测试结果与标准输出进行对比
- 异常情况的分析处理：对不符合预期的测试结果的分析、记录、分类和报告等
- 总体测试状况的统计及报表的产生
- 自动化开发测试流程：如自动化测试与开发中产品**每日构建**的配合等



自动化测试工具的常见功能

- 其它功能

- 确定系统最优的硬件配置
- 检查系统的可靠性
- 检查系统软硬件环境
- 模拟各种设备
- 监控软硬件系统