Pywinauto 在 Windows Twain Driver 自动化测试中的应用研究

翁省辉[1]，喻武龙[2]
（1.北京理工大学 珠海学院 计算机学院，广东 珠海 519088；
2.北京理工大学 珠海学院 信息学院，广东 珠海 519088）

摘 要：以 Python 为基础，结合对 Twain Driver 测试工具的具体需求，将 Pywinauto 引入到 Twain Driver 的自动化测试中。介绍了 Pywinauto 的基本概念，通过测试用例说明 Pywinauto 在自动化测试中的具体实现。应用结果表明，该方法大大提高了测试的自动化程度，极大地减少了 Twain Driver 测试的工作量，同时也确保了测试质量。

关键词：Python；Pywinauto；Twain Driver；自动化测试
中图分类号：TP311.56 **文献标识码**：A **文章编号**：1674-7720(2012)03-0008-03

Application research of Pywinauto in the Windows Twain Driver automation testing

Weng Shenghui[1], Yu Wulong[2]
(1.School of Computer, Zhuhai Campus, Beijing Institute of Technology, Zhuhai 519088, China;
2.School of Information, Zhuhai Campus, Beijing Institute of Technology, Zhuhai 519088, China)

Abstract: In this paper, Pywinauto will be introduced to Twain Driver automation testing based on Python and special needs for Twain Driver test tools. This paper firstly introduces the basic concepts of Pywinauto, and then introduces how Pywinauto is used in the implementation of automated testing by a test case. The test results show that this solution can promote the degree of automation greatly, and greatly reduce the workload of Twain Driver test, at the same time it can ensure the test quality.

Key words: Python; Pywinauto; Twain Driver; automation testing

Windows 下扫描仪驱动程序主要使用 Twain 协议[1]。由于 Twain 协议的复杂性，Twain Driver 一般会以图形界面方式提供众多扫描选项以供用户使用。扫描选项数量众多以及选项之间的依赖性，决定了 Twain Driver 的测试是一项非常艰巨的任务。Twain Driver 的测试主要包括基本功能测试以及回归测试。一般一个 Twain Driver 包含近百个基本功能测试用例以及随着缺陷数量增长而不断增加的回归测试用例。特别是临近产品正式发布日期时，每修正一个缺陷，都会带来极大的工作量：一方面，要做基本功能测试检查是否有新的缺陷；另一方面，要做回归测试检查之前已修正好的缺陷是否受到影响。由于此时产品已处于开发周期的后期阶段，缺陷的基数通常会比较大，回归测试的测试用例将变得非常庞大。更重要的是，如果有多个缺陷需要修正，那么以上两个方面的测试还将重复执行多次，测试工作将成倍增长。

针对以上问题，本文以 Python 为基础，结合对 Twain Driver 测试工具的具体需求，提出了一个基于 Pywinauto 实现自动化测试的解决方案[2]。Pywinauto 通过模拟测试人员在用户界面上的鼠标、键盘操作，来减少测试人员的手工操作。应用结果表明，该方案能够极大地提高测试效率，在减少了测试时间的同时，也确保了产品质量。

1 Pywinauto 的基本概念

Pywinauto 是基于 Python 开发的，用于自动化测试的脚本模块的第三方扩展包，它通过向 Windows 对话框和控件发送鼠标、键盘动作来实现 Windows 图形界面的自动化测试[3]。

1.1 标识应用程序实例

Pywinauto 在使用前首先需要将应用程序实例连接到一个进程，有两种标识方法分别对应两种情况：

（1）应用程序未启动即应用程序实例不存在：此时可以调用 start_（self，cmd_line，timeout=app_start_timeout）来启动应用程序。示例如下：

```
gAppName =ur"C:\\Program Files\\TWAIN Working Group\\TWAIN Toolkit\\Twack_32.exe"
app=application.Application().start_(gAppName)
```

（2）应用程序已启动：此时只需调用 connect_（self，**kwargs）连接到已运行的应用程序。示例如下：

```
AppName=ur"TWAIN_32 Twakcer"
app=application.Application().connect_(title_re= AppName)
```

1.2 标识应用程序窗口

在取得应用程序实例之后，就可使用该实例标识应用程序窗口，主要有 3 种标识方式：

（1）使用窗口标题。示例如下：

```
gWizardName="Select"
MainDlg=app[gWizardName]
```

或者将窗口标题直接当成一个变量形式使用，但这样标识，非英文语系时窗口会出现问题，所以这种方法并不推荐。示例如下：

```
MainDlg=app.Select
```

（2）窗口标题结合正则表达式，特别是当窗口标题不确定或经常变化时尤为有用。示例如下：

```
dlg=app.window_(title_re=".*doc",class_name="#33888")
```

（3）直接取最上层窗口。此时需要确保被标识的应用程序窗口为顶层窗口。示例如下：

```
MainDlg=app.top_window_()
```

1.3 标识应用程序窗口控件

Pywinauto 的测试原理主要模拟控件上的手工操作，所以 Pywinauto 自动化测试中重要的一环就是标识应用程序窗口上的控件。假设应用程序窗口有一内容为 OK 的 Button 控件，主要有两种方式标识这个控件：

（1）使用窗口控件标题。示例如下：

```
app["dlg"]["OK"]
```

或者将控件标题直接当成一个变量形式使用。与标识窗口类似，这种方法也不推荐。示例如下：

```
app.dlg.OK
```

（2）使用 Friendly class，特别是当控件标题内容为空时尤为有用。示例如下：

```
Dialog.Button1
```

需要说明的是：Button、Button0 及 Button1 都是代表第一个 Button，Button2 代表第二个 Button。标准控件通过 Friendly class 很容易标识出来，但是非标准控件其 Friendly class 并不明显，这时可通过 Visual Studio 自带的 Spy++来查看，如图 1 所示。

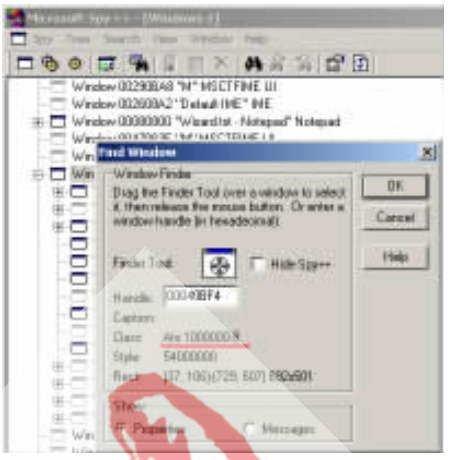

图 1 使用 Visual Studio 自带的 Spy++查看 Friendly class

1.4 如何操控鼠标与键盘

在取得 Windows 对话框和控件后，就可以向对话框或者控件发送鼠标、键盘操作来实现自动化测试。

鼠标的操作：（1）点击操作：模拟鼠标的点击操作可以结合具体的控件，Pywinauto 对于不同的控件提供了不同的函数。例如，模拟 Next Button 的点击事件可以表示为 Dialog.["Next"].Click()；模拟 ComboBox 控件的选择操作可以使用如下方式：Dialog.ComboBox1.Select（1）。（2）拖动操作：主要使用 PressMouse()、MoveMouse() 和 ReleaseMouse() 实现鼠标的按下、移动以及释放操作。示例如下：

```
def AdjustSize (WizardDlg, ShiftX, ShiftY):
    Offset=10
    OrgRect=WizardDlg.WiaControl1.Rectangle()
    WizardDlg.WiaControl1.PressMouse(coords=(Offset,Offset))
    WizardDlg.WiaControl1.MoveMouse (coords=(Offset+ShiftX, Offset+ShiftY))
    WizardDlg.WiaControl1.ReleaseMouse()
```

按键的操作：Pywinauto 使用 SendKeys 来进行按键处理[4]。一些程序并不会将菜单项指定给主 UI（如 Word），这样就不能直接使用菜单方法，而是使用快捷键的方式，这就需要使用 SendKeys 发送快捷键。如要表示按下 Alt+F 组合键，可以写成 MainWin.TypeKeys("%F")。需要说明的是：TypeKey 还可表示按照一定时间间隔接受多个组合按键。例如，在 Word2003 打开从扫描仪导入图片的窗口，需要先按 Alt+I，然后按 Alt+P，最后按 Alt+S。用 SendKeys 可以表示为：

```
MainWin.TypeKeys("%IPS",pause=0.5)
```

1.5 中文支持

Pywinauto 在对中文应用的菜单进行操作时，通常会由于编码问题而使中文应用的对话框和控件无法进行标识。可以使用以下两种方法解决：

（1）使用"u"或者"ur"将字符串转换成 UTF 格式的字符串。例如：

```
gDialogName=u"选择来源"
gButtonName=u"选定"
app[gDialogName][gButtonName]
```

（2）使用 decode 函数强行转换字符串编码。例如：

```
CP="cp936"
gDialogName="选择来源".decode(CP)
gButtonName="选定".decode(CP)
app[gDialogName][gButtonName]
```

2 Twain Driver 自动化测试实现

由于 Twain 协议使用的广泛性，目前已经有很多应用程序支持该协议。Windows 下常见的应用程序主要有 Twack_32、Microsoft Word、PageManager 以及 Adobe Photoshop 等，这些应用程序均可作为 Twain Driver 的测试工具。其中 Twack_32 是 TWAIN 官方所提供的一个工具，其兼容性最好，而且它不仅提供了 TWAIN 应用的例程，还可以在计算机系统上安装一个虚拟的图像输入设备（TWAIN_32SampleSource），所以测试人员通常使用 Twack_32 对 Twain Driver 进行测试。本文也以 Twack_32 为例实现 Twain Driver 的自动化测试。

2.1 Twack_32 启动的实现

下载并安装完后，打开 Twack_32 界面，然后依次选择 File->Select Source，弹出一个对话框，如图 2 所示。

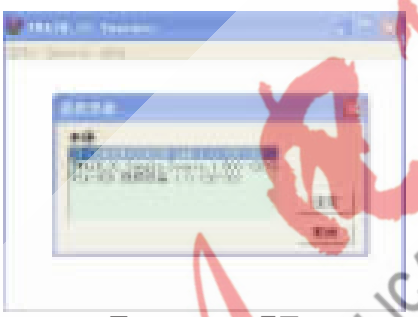

图 2 Twack_32 界面

实现时，首先通过调用 start 函数启动应用程序获取应用程序实例，然后利用该应用程序实例和界面的标题（TWAIN_32 Twacker）取得应用程序窗口实例，最后根据应用程序窗口实例取得控件标识，之后就可以操纵该控件了。中文菜单可使用"u"转换字符串。实现代码如下：

```
AppName='TWAIN_32 Twacker'
TWAINDS_NAME='SP C240SF/C242SF LAN 0.59 (32-32)'
def RunTwack():
    app=application.Application()
    app.start_ (ur"C:\\Program Files\\TWAIN Working Group\\TWAIN Toolkit\\Twack_32.exe")
    app[AppName].Wait('ready')
    app[AppName].MenuSelect("File->Select Source...")
    app[u'选择来源'].Listbox1.Select(TWAINDS_NAME)
    app[u'选择来源'][u'选定'].Click()
    return app
```

2.2 自动化测试实现

Twack_32 成功启动后，如果安装了需要测试的驱动，将会出现被测 Twain Driver 的界面，图 3 为某一驱动的界面，以下实现也以此驱动为例。

图 3 某一驱动程序的界面

该驱动支持三种扫描模式，分别是：Full Color、Gray 和 Black and White。假设需要测试该驱动的扫描模式是否实现正确，首先需要将 mode 改为 Full Color，其他参数不变，扫描并观察图片是否正确；然后依次将 mode 改为 Gray 和 Black and White 并重复以上操作。使用 Pywinauto 实现对扫描模式的自动化测试。实现时尽量将每一个测试案例用一个相应的函数实现，有利于测试脚本的后期维护。示例代码如下：

```
DriverName='SP C240SF/C242SF LAN'
def SetScanParameter(ScanApp,nPaperSource,nColorModelItem,nDPIItem,nSizeIndex):
    WaitForWindowIsReady(ScanApp[DriverName])
    ScanApp[DriverName].ComboBox.Select(nPaperSource)
    WaitForWindowIsReady(ScanApp[DriverName])
    ScanApp[DriverName].ComboBox2.Select(nColorModelItem)
    ScanApp[DriverName].ComboBox3.Select(nDPIItem)
    ScanApp[DriverName].ComboBox4.Select(nSizeIndex)
    ScanApp[DriverName].Scan.Click()
    time.sleep(0.5)
def ScanImageWithTwackChangeMode(ScanApp,nColorModelItem)
    SetScanParameter(ScanApp,1,nColorModelItem,1,2)def TestColorMode(ScanApp)
    ScanImageWithTwackChangeMode(ScanApp,1)
    ScanImageWithTwackChangeMode(ScanApp,2)
    ScanImageWithTwackChangeMode(ScanApp,3)
```

本文以 Twain Driver 为例介绍了 Pywinauto 在图形界面自动化测试中的应用与实现。Pywinauto 通过模拟测试人员在用户界面上的鼠标、键盘操作来减少测试人员的手工操作。实现时，将各个测试用例对应一个函数，然后根据测试需求调用相应的函数。由于基本功能测试时测试用例相对稳定，所以脚本一旦写好，以后每次发布版本前只需运行一次脚本即可完成基本功能测试。而在回

归测试阶段，每增加一个测试用例，便增加一个相应的实现函数，从而避免遗漏对之前版本缺陷的测试。此外，本文的方案只需稍作修改，便可应用于其他 Windows 图形应用程序的测试，特别是程序界面手动操作比较复杂时，该方案的效果更加明显。

参考文献

[1] TWAIN—standard for image acquisition devices[DB/OL]. http://twain.org.2001.

[2] 辛敏杰，高建华.一种改进的 GUI 测试框架 DART[J].计算机工程，2009，35(7)：55-58.

[3] Contents -pywinauto v0.4.1 documentation [DB/OL]. http://pywinauto.googlecode.com/hg/pywinauto/docs/contents.html,2010.

[4] SendKeys |Rutherfurd.net [DB/OL]. http://www.rutherfurd.net/python/sendkeys,2008.

（收稿日期：2011-09-06）

作者简介：

翁省辉，男，1984 年生，在读硕士，主要研究方向：嵌入式系统移植与驱动开发。

喻武龙，男，1979 年生，讲师，主要研究方向：嵌入式系统移植与驱动开发。