基于.NET 平台 GUI 自动化测试框架的设计

段 莹,郭利刚

(武汉理工大学 计算机学院,湖北 武汉 430063)

摘 要:分析了录制回放技术的基本原理和缺陷,运用 WIN32API 和.NET 反射机制,设计了一个改进的轻量级 GUI 自动化测试框架,解决了当前测试自动化中出现的一些棘手问题。

关键词:GUI;自动化测试;反射

中图分类号:TP311.5 **文献标识码**:A **文章编号**:1674-7720(2010)19-0004-03

Design of GUI test automation framework based on .NET platform

DUAN Ying, GUO Li Gang

(School of Computer Science and Technology, Wuhan University of Technology, Wuhan 430063, China)

Abstract: This paper analyzed the principle and flaws of the traditional automated test tools, applied .NET reflection and WIN32 API technologies, designed and implemented an improved lightweight GUI automated testing framework that can help users avoid the flaws emerged in the current test tools.

Key words: GUI; automated test; reflection

软件测试是保证软件质量的有效手段。目前,在 GUI 自动化测试中,很多软件体系都采用录制回放技术。这种技术要求测试者通过鼠标和键盘的点击进行工作,脚本记录事件,然后以自动化测试的方式进行回放。记录下来的测试脚本必须经过编辑和调试之后插入验证和检查点。产生的脚本通常是硬编码,需要测试人员对脚本进行编辑以及参数化操作。同时,界面元素属性的任何变化都会影响脚本的运行,有时甚至需要重新录制脚本。

1 .NET 中的反射机制

通常,应用程序(包括桌面程序和 Web 应用)都由一些基本的界面控件组成,所有的软件指令都是通过控件以事件或消息的形式传递给后台处理。GUI 自动化测试的本质是对 GUI 中的控件元素提供编程手段[2]。在基于 GUI 对象识别和控制的自动化测试工具中,过去一直依赖于 Windows API 函数的调用。而随着新的编程语言和平台的出现,涌现了很多新的语言特性,这些语言特性可用于自动化测试工具的设计,例如反射机制就是其中一项技术。

反射(Reflection)是.NET 中的重要机制,通过反射可以在运行时获得.NET 中每一个类型(包括类、结构、委托、接口和枚举等)的成员,包括方法、属性、事件及构造函数等,还可以获得每个成员的名称、限定符和参数等。如果获得了构造函数的信息,即可直接创建对象,即使这个对象的类型在编译时还不知道。程序集包含模块,而模块包含类型,类型又包含成员,反射则提供了封装程序集、模块和类型的对象。可以使用反射动态地创建类型的实例,将类型绑定到现有对象或从现有对象中获取类型,然后调用类型的方法或访问其字段和属性[3]。

2 框架的整体设计

自动化测试框架的搭建基本上占了整个自动化测试工作量的 40%,是自动化测试实施的一个重要组成部分。软件自动化框架从本质看是一系列的策略思想、规范文件和代码的集合。本文提出一种改进的轻量级的 GUI 自动化测试框架,该框架可以帮助用户避免当前测试工具出现的缺陷。此框架将具备以下五个特性:GUI 控件自动搜索、自动生成和执行测试脚本、基于数据驱动的原则、测试的自动验证、使用编程语言开发。框架的整体设计如图 1 所示。从图 1 可以看出,该框架让测试人员从繁重的录制工作中解放出来,将更多的时间和精力集中在测试用例的设计中。

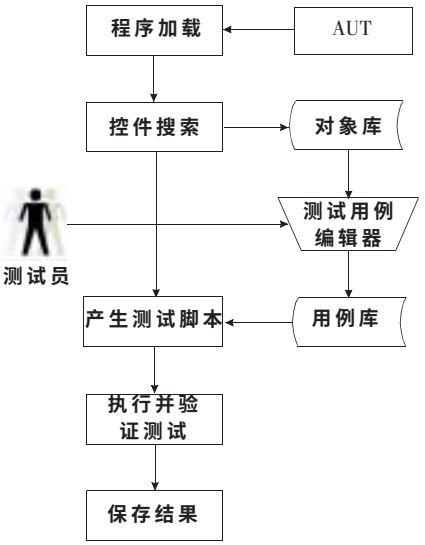

图 1 框架的整体设计

3 框架具体开发

3.1 加载被测试程序

为了测试 GUI 应用程序，必须在测试工具中运行被测程序使两个程序交互。使用 Assembly 定义和加载程序集，加载在程序集清单中列出模块，并从此程序集中查找类型，创建该类型的实例。为了使这两个应用程序实现交互，必须通过多线程机制实现。下面是通过反射加载被测程序的核心代码[4]：

```
Assembly asm=Assembly.LoadFrom(path);
Type t1=asm.GetType(formName);
testForm=(Form)asmCreateInstance(t1.FullName);
ParameterizedThreadStart pt=new ParameterizedThreadStart
(AppRun);
Thread thread=new Thread(pt);
thread.Start(testForm);
private void AppRun(Form theForm)
{
    Application.Run(theForm);
}
```

3.2 GUI 控件搜索

大部分自动化功能测试工具，尤其是商业的测试工具，都是基于 GUI 对象识别技术设计的。基本思想是每个基于窗体的控件都是一个窗体，每个控件或窗体都有一个句柄来进行访问、操作和检查。

实现 GUI 测试自动化的困难之一是测试工具并不知道被测程序中存在哪些 GUI 部件。录制回放工具使用手工录制过程暂时解决了这一问题。Win32 API 中封装了很多可用于自动化测试编程的函数，这些函数可在编程语言进行调用，实现自动化测试编程。本文的自动化 GUI 测试工具将采用 Win32 API 对被测程序进行自动、系统、全面的控件搜索。实现该搜索将用到 Win32 API 中封装的可用于自动化测试编程的函数，包括：GetWindowRect、mou se_event、GetCursorPos 和 WindowFromPoint 函数等。GetWindowRect 函数返回指定窗口的边框矩形的尺寸。该尺寸以相对于屏幕坐标左上角的屏幕坐标给出。通过使用这个函数可以计算出窗口的宽度和高度。mouse_event 函数能模拟鼠标击键和鼠标动作。GetCursorPos 函数检取光标的位置，并以屏幕坐标来表示。使用 WindowFromPoint 函数能获得包含指定点的控件的句柄。一旦得到了窗口的句柄，就能得到控件的文本、类名以及父窗口的句柄。为了对界面进行彻底的控件搜索，该框架将使用嵌套循环，从界面的左上角到界面的右下角依次移动鼠标进行控件识别，并将结果保存到对象库中[5]。

该模块实现的伪代码如下：

```
RECT rt=new RECT();
GetWindowRect(iHandle, ref rt);
width=rt.Right-rt.Left;                    //得到界面的宽度
height=rt.Bottom-rt.Top;                   //得到高度
step=8;                                    //鼠标每次移动的像素
for (int x=0;x<width;x+=step)
{
    for (int y=0;y<height;y+=step)
    {
        mouse_event();                     //移动鼠标到相应的坐标
        GetCursorPos();                    //得到坐标点处的光标
        WindowFromPointGet();              //得到此处的控件句柄
        GetWindowText();                   //得到窗体的文本
        GetClassName();                    //得到控件的类别
        GetParent();                       //得到父窗体的句柄
        if (the handle does not exist in object repository)
        then save the infomation.
    }
}
```

通过对界面的彻底搜索，可以得出控件的句柄、文本、类名、父窗体的句柄以及 GUI 控件间的层次关系。

3.3 生成测试用例

实现 GUI 测试自动化的另一个问题是测试工具不能按事件发生的顺序来选择和操作控件。传统的测试工具通过录制的方法记录程序运行的顺序，但是这种机制存在很多限制。本文设计的自动化测试框架的思路是经过界面控件的彻底搜索后，控件的详细信息会保持到对象库中。此时，测试人员可以通过测试用例的输入/输出模块来编写测试用例。该模块是一个可视化编辑器。测试用例编辑器从对象库中导入对象信息，该模块根据测试用例设计人员的操作顺序自动产生事件的顺序并将测试步骤保存到 XML 文件中。按事件发生的顺序存储的相关事件可以形成一个测试场景。测试用例编写者可以对产生的测试场景进行编辑，改变测试步骤的顺序或

者添加更多的事件。测试运行模块将会从 XML 文件中读取测试用例，按事件的顺序执行相应的操作。

3.4 执行并验证测试

基于数据驱动的原则，在测试用例产生之后，测试用例文件驱动测试的执行。不同的控件类型将从测试类库中调用不同的测试执行方法。该模块采用反射机制来模拟用户的操作。反射提供了延迟绑定，能模拟用户的常用操作，如编辑控件文本、单击按钮等。调用控件的事件方法的关键是要用到反射中的 MethodInfo.Invoke 方法。如果没有潜在的其他问题，可以像下面这样调用控件的事件方法：

```
Type t1=testForm.GetType();
MethodInfo mi=t1.GetMethod("button1_Click",flags);
mi.Invoke(testForm,new object[]{null,EventArgs.Empty});
```

注意在调用该方法时要充分考虑线程的问题。该方法不是从被测应用程序运行的线程被调用的，而是从测试工具本身的主线程中被调用的。因此，应该通过调用 Form.Invoke 方法间接运行 MethodInfo.Invoke 方法[4]。

验证测试过程中，为了得到控件的属性或字段信息，将用到反射中的 Form.GetType、Type.GetField、FieldInfo.GetValue、PropertyInfo.GetValue 等方法。通过比较测试用例的预期结果和实际运行结果来决定测试结果。然后将测试用例的编号、预期结果、实际结果以及时间信息保存到 XML 文件中，同时可采用 XSLT 方式将测试结果以更友好的方式展现出来。

3.5 使用 Windows 调度程序

测试人员可能希望测试工具能在指定时间自动地运行测试，例如晚上 11:00 点。这样，整个测试工作可以在下班之后进行，到第二天就能看到测试运行的结果。事实上，Microsoft 都包含一个能在给定时间执行命令的程序。当前版本中，这个程序是 Scheduled Tasks，位于控制面板上。打开这个文件夹，会看到 Add Scheduled Task 图标，单击它可以启动向导。有一个重要的选项是 Task 选项卡上的 Run 文本框，如图 2 所示。可以在该文本框中添加相应的参数来完成自动化测试[6]。

本文设计了一个轻量级的 GUI 自动化测试框架，该

图 2 Scheduled Tasks 选项卡

框架可以系统、全面、主动地实现控件搜索，避免了传统测试工具中繁重的录制过程。界面控件的任何改变也不会影响测试脚本的运行。该框架让测试人员从繁琐的录制工作、测试脚本的编辑和验证点的插入工作中解放出来，使测试人员能将更多的时间和精力集中到测试用例的设计中。

参考文献

[1] BECK K. Test driven development: by example [M]. Addison-Wesley Professional, 2002.

[2] DUSTIN E, RASHKA J, PAUL J. Automated software testing introduction, management, and performance [M]. Boston addison Wesley, 1999.

[3] 柳胜.软件自动化测试框架设计与实践[M].北京：人民邮电出版社，2009.

[4] ZHOU R Z, WEI Z K. Study on Application of .NET Reflection in Automated Testing [J]. ICACT, 2010,7-10:(797-800).

[5] MCCAFFREY J D. .NET test automation recipes: a problem-solution approach[M]. Apress, 2006.

[6] LI K L, WU M Q. Effective GUI testing automation: developing an automated GUI testing tool[M].王轶昆，译.北京：电子工业出版社，2005.

[7] ROOT R, SWEENEY M R. A tester's guide to .NET Programming[M].杨浩，译.北京：清华大学出版社，2007:279-280.

(收稿日期：2010-06-13)

作者简介：

段莹，女，1987 年生，硕士研究生，主要研究方向：软件工程、软件测试自动化。

郭利刚，男，1985 年生，硕士研究生，主要研究方向：软件工程、信息检索。