一种基于 GUI 的测试脚本开发环境

A GUI Based Development Environment for Test Scripts

阚红星 （中国科学技术大学软件学院 230026）
龚育昌 （中国科学技术大学计算机系 230026）

摘要：为了提高自动化测试的效率，本文提出了一种基于 GUI 测试脚本的开发环境。该环境以各种 GUI 动作为核心，形成可重用的动作类库，同时利用外观模式（facade）为类库提供一个简单接口。这样，测试者在设计脚本时只要利用这个简单接口去实例化动作类库就可以了，而不必做复杂的脚本编程工作。

关键词：脚本测试 可重用性 动作类库 GUI 接口

1 引言

为了保证软件的质量，在开发软件的过程中，我们必须进行独立的软件测试，且这种测试需要贯穿整个软件开发的始终，而不仅仅是最后软件的确认和验证。

国外软件测试的发展已趋于成熟。在软件业较发达的国家，软件测试不仅早已成为软件开发的一个有机组成部分，而且在整个软件开发的系统工程中占据着相当大的比重。如微软的开发工程师与测试工程师的比例是 1∶2，而我国则是 6∶1。在自动化测试方面，国外已经有许多成熟的可应用的测试自动化工具，而我国这方面则刚刚起步。其实，随着软件业的快速发展，自动化测试的作用会越来越明显，因为：

（1）自动化测试可以极大地提高测试的质量和效率，增加软件的可信度。

（2）自动化测试可以减少测试过程中的重复劳动，自动地、频繁地测试以降低测试成本。

（3）自动化测试可以替代手工测试的困难，如并发测试、大数据量测试、崩溃性测试等，用人来测试是不可能达到的。

本文以各种“GUI”动作为核心，提出一种基于 GUI 自动化测试脚本的开发环境，该环境简单、易用、具有可重用性和一般性，可以极大地提高自动化测试的效率。

2 影响自动化测试效率相关因素

那么影响自动化测试效率的因素有哪些呢？

首先，编写测试脚本的开销影响自动化测试的效率。编写测试脚本的开销越大，自动化测试的效率越低。快速地开发出测试脚本无疑会提高自动化测试的效率。

其次，测试脚本的可重用性也是影响自动化测试的效率的重要因素。手工测试的过程常分下列几个阶段：制定测试计划，设计测试用例，手工执行测试，评估测试结果，而自动化测试则有一个测试脚本的编写和维护过程。一般情况下，脚本的编写和维护时间要远远大于一次手工测试的时间。如果测试脚本在测试被测程序时只运行了一次，那么自动化测试的代价要远远大于手工测试的代价，也就是说自动化测试将毫无意义。所以，编写和维护脚本的开销只能在脚本的重复执行中得到回收。实践证明，随着测试脚本执行次数的增加，自动化测试的效率会越来越高，最终将会远远超过手工测试。要使测试脚本具有高度的重用性，只有增加测试脚本的技术含量，使其尽可能多的适应各种测试环境而具有一般性。

3 基于 GUI 测试脚本的开发环境及其设计

3.1 基于 GUI 测试脚本的开发环境

基于 GUI 的测试无非是测试 GUI 对象（如按钮、文本框、下拉菜单等）是否能正确地接受事件的响应并完成指定的功能。也就是说整个测试过程就是执行一系列相关的鼠标和键盘动作以检查软件是否能正常运行。

下面将以测试过程中各种 GUI 对象的动作为核心，提出一个基于 GUI 测试自动化脚本的设计环境。该环境的结构如图 1 所示：

自动化测试人员首先根据测试用例抽取出相关待测试的 GUI 动作，然后利用测试工具（如 Rational Robot，LD，TD 等）分析相关 GUI 动作的路径。最后，测试者可以用具体路径下的动作，通过抽象接口去实例化动作类库。这样，测试人员就可以在较短的时间，开发出具有高度重用性的测试脚本，从而提高自动化测试的效率。

3.2 抽象接口的设计

在动作类库中存放着各种对 GUI 对象的操作实现细节。一般情况下，为了使动作类库具有很高的重用性，需要尽可

能多地收集 GUI 对象,从而使动作类库越来越庞大,使得函数之间的交叉调用关系变得复杂而难以维护。另外,客户程序的直接实例化,会使得客户程序与抽象类的实现部分存在很大的依赖性。由于调用模块和被调用模块之间不具有低耦合性,从而降低了系统的独立性和可移植性。

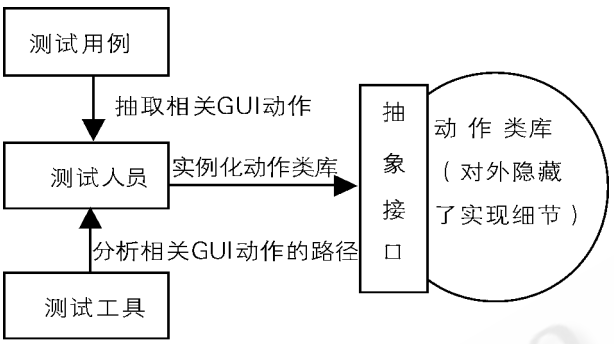

图 1　基于 GUI 自动化测试脚本的开发环境

本文提出的抽象接口可以解决上述问题,所谓抽象接口是利用了设计模式中的外观模式(fa? ade)。它为复杂的动作类库提供了一个对外的规范接口,同时也降低了用户程序和动作类库之间的耦合性。

通过外观模式设计的动作类库统一接口由关键字 interface 来定义,这个接口中包含着一系列具有公共属性的抽象方法。外界的任何实例化只与接口打交道,而不直接与类库发生联系。当外界的对象要使用类库中的模块以完成某一动作时,它首先将对象的消息传递给接口,由接口按照动态绑定的机制把该对象与其具体实现的细节联系起来。下面是实现接口的说明模式:

```
    // Action 即代表抽象的统一接口
public interface Action{
        public abstract TextAction(……);
        public abstract ButtonAction(……);
        public abstract CheckBoxAction(……);
        public abstract CombBoxAction(……);
        ……
}
```

3.2 动作类库的设计

动作类库的设计是测试脚本设计的核心。对共享的、可重用的动作类库来说,需要把具体的 GUI 动作封装在某个类中,然后在实例化过程中调用它。根据常用的 GUI 动作的类型,可以把它们分为文本框类(TextObject)、按钮类(ButtonObject)、复选框类(CheckBoxObject)、下拉列表类(ComboBoxObject)等等。在每个类中都封装了相应的动作实现方法。例如动作函数 ButtonAction(……)就封装在 ButtonObject 类中,以实现具体操作按钮的动作。下面是几个典型的类的设计过程:

文本框类(TextObject)的设计:

```
public class TextObject implement Action{
        private string path; //动作的路径
        private EditBox box; //动作的类型为文本框
        private string str; //文本框中的值
    public TextObject(string p, EditBox b,string s ) //有参构造函数
            { path=p; box=b; str=s;}
    public string getpath( ) { return path } //得到私有变量的值
    public EditBox getbox( ) {return box }
    public string getstr( ) {return str }
    public void TextAction( ){
    SQASetProperty getpath( ), getbox( ), "value", str; //在文本框中输入内容
}
```

按钮类(ButtonObject)的设计:

```
public class ButtonObject implement Action{
        private string path;
        private Button bton;
    public ButtonObject(string p, Button b ) //有参构造函数
            { path=p; bton=b;}
    public string getpath( ) {return path} //得到私有变量的值
    public Image getbton( ) {return bton}
    public void ButtonAction( ){ //得到路径和要操作的对象并进行操作
            getpath( );
            getbton click; //对按钮进行操作
    }
```

下拉列表类(ComboBoxObject)的设计:

```
public class ComboBoxObject implement Action{
        private string path;
        private ComboBox cbb;
public ComboBoxObject(string p, ComboBox b ) //有参构造函数
            { path=p; cbb=b;}
  public string getpath( ) {return path} //得到私有变量的值
  public ComboBox getcbb( ) {return cbb}
```

```
public void ComboBoxAction( ) {
    SQAGetProperty getpath( ) ,getcbb( ) , "indexMax" , index ; //取得下拉列表中列表相的最大数目
    ObjectIndex =1 + ( int) ( math. random * index) ; //随机地取出一个列表相以供操作
    ComboBox click, ObjectIndex; //操作相应的列表相
}
```

同样,可以设计其他的测试 GUI 的动作类。

动作类库设计好后,测试者在此基础上编写可重用性的测试脚本就简单多了。在主模块中,测试者只要让具体的类对象去实例化抽象的类就可以了。其算法如下:

```
main{
  //生成一个具体的 Button 对象
  ButtonObject ButtonObject = new ButtonObject ( path,bton)
  //定义 point 为抽象接口 Action 类
  Action point;
  //让抽象接口指向具体的类对象,这是实现多态性的必要步骤
  Point = ImageObject;
  ……; }
```

如下图 2 所示:主模块实例化动作类库的实现机制:

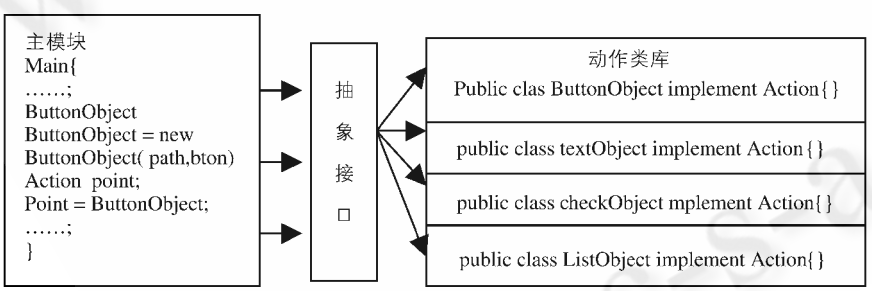

图 2　主模块实例化动作类库的实现机制

4　基于 GUI 自动化测试脚本开发环境的应用

下面通过一个"网络培训系统"的示例来说明如何利用上面的脚本开发环境来开发一个具有高度重用性的自动化测试脚本。

4.1　网络培训系统的需求和测试用例

现有一个基于 WEB 的网络培训系统。该系统具有如下功能:主持人可以打开培训中心的软件,注册安排一个培训,并可以对培训的时间,参加培训的人数以及该培训的类型进行设置。当然,主持人还可以根据一些特殊需要管理和设置培训系统。该系统还具有会议帮助系统,以帮助、支持和培训参训人员。主持人主持一个培训后,别人可以在异地登陆以参加这个培训并能正常结束。

根据上面的需求,可以设计出以下一个具有代表性的测试用例,该测试用例测试基本的主持培训功能,包括以下序列活动:

- 用浏览器打开培训中心
- 按"Image"注册,输入帐号和密码
- 按"login"登陆培训中心
- 输入培训主题、密码
- 按"Schedual"按钮并主持会议

4.2　分析测试用例中相关的 GUI 动作和路径

从上面的测试用例中可以得到以下的键盘输入、鼠标点按等动作:

(1) 打开网络培训系统
(2) 点按注册的"Image"按钮进行注册
(3) 在文本框中输入帐号和密码
(4) 点按登陆的"login"按钮进行登陆
(5) 在文本框中输入培训主题、密码
(6) 点"Schedule Training"进行链接
(7) 点"Schedule"按钮安排一个培训

在这个流程中,包含了六个相关 GUI 的动作。在上面的可重用的动作类库中,已经有了这六个动作的具体实现过程。

让可重用的类库去执行具体的 GUI 操作时,我们要告诉类库它执行的是哪一个页面上的 GUI 动作。例如相同的按钮动作可能在不同的页面上出现,那么如何区别它们呢? 可以"路径"对它们进行区别,不同的路径即代表不同的页面。其实,要发现 GUI 动作所在的路径很容易。如通过测试工具的录制就能很快发现 GUI 动作的路径。

通过简单的录制可以得到以下动作的路径:

"Image"的路径:HtmlFrame;\;main;\;TopFrame
帐号和密码的路径:HtmlFrame;\;main;\;HostLog
"Login"的路径:HtmlFrame;\;main;\;HostLog
培训主题和密码的路径:HtmlFrame;\;left;\;Menu
"Schedule Training"的路径:HtmlFrame;\;left;\;Menu
"Schedule "的路径:HtmlFrame;\;Botton

4.3　设计测试主模块

测试的动作类库设计好以后,测试者只要用一个简单的主模块实例化动作类库就可以了,而不必要做一些复杂的编程工作。

```
main{
//直接输入网址打开网络培训系统
InputKeys http://regions.webex.com{ENTER}
//点按注册的"Image"按钮进行注册
ButtonObject ButtonObject = new ButtonObject (HtmlFrame,
\,main,\,TopFrame ,Button)
  Action point1;
  Point1 = ButtonObject;
//在文本框中输入帐号和密码
TextObject TextObject = new TextObject (HtmlFrame, \,
main,\,TopFrame ,EditBox, "admin")
    Action point2;
    Point2 = TextObject;
TextObject TextObject = new TextObject (HtmlFrame, \,
main,\,TopFrame ,EditBox, "123456")
    Action point3;
    Point3 = TextObject;
//点按登陆的"login"按钮进行登陆
ButtonObject ButtonObject = new ButtonObject (HtmlFrame,
\,main,\,HostLog ,Button)
    Action point4;
    Point4 = ButtonObject;
//在文本框中输入培训主题和密码
TextObject TextObject = new TextObject (HtmlFrame, \,
main,\,TopFrame ,EditBox, "meeting")
    Action point5;
    Point5 = TextObject;
TextObject TextObject = new TextObject (HtmlFrame, \,
main,\,TopFrame ,EditBox, "123456")
  Action point6;
  Point6 = TextObject;
//点"Schedule Training"进行链接
ComboBoxObject ComboBoxObject = new ComboBoxObject
(HtmlFrame,\,left,\,menu ,ComboBox)
  Action point7;
  Point7 = ComboBoxObject;
//点"Schedule"按钮安排一个培训
ButtonObject ButtonObject = new ButtonObject (HtmlFrame,
\,bottom ,Button)
  Action point8;
  Point8 = ButtonObject;
}
```

从上面的主模块中可以看到,尽管动作类型不同,但实例化过程基本类似,没有什么复杂的编程工作,编写起来快速、方便。

5 结束语

自动化测试同手工测试比较起来其主要的开销来自测试脚本的设计和编写。因此,缩短测试脚本的编写时间,增加测试脚本的技术含量是提高自动化测试效率的关键所在。本文以快速开发具有高度重用性和共享性的测试脚本为出发点,提出一种基于 GUI 自动化测试脚本的开发环境。该环境以测试过程中各种 GUI 对象的动作为核心,形成动作类库,同时结合设计模式中的外观模式(fa? ade)对外提供一个简单接口。这样,测试者在设计脚本时只要简单地实例化动作类库就可以了,而不必要做复杂的编程工作。

当然,该环境也有不足之处,如动作路径识别需要借助其他测试工具才能完成。因此,下一步的工作将把动作路径识别这内容也加入到类库之中,使该环境完全脱离其他测试工具,成为简单、易用、独立的基于 GUI 的测试工具。

参考文献

1 张海藩,软件工程导论[M],清华大学出版社,2002。

2 丁玉伟,GUI 自动化测试收益回收期分析[EB/OL]. http://www.horizon-biz.com/xuqiu.htm。

3 Erich Gamma, Richard Helm, Ralph Johnson. Design Patterns[M]. Addison Wesley Longman, Inc. 2000. 121-128.

4 Harvey M. Deitel, Paul J. Deitel. Java How to Program, Third Edition[M]. Prentice Hall, Inc. 2001. 319-367.