一种关键字驱动的 Java API 自动测试框架

高青娟,陈立潮,张英俊,潘理虎,谢斌红
(太原科技大学 计算机科学与技术学院,山西 太原 030024)

摘 要:传统的对应用程序编程接口的测试需要编写测试驱动程序驱动被测 API,然后编译驱动程序,接着执行驱动程序,最后要对测试运行结果做分析。过程繁琐而且对测试人员的编码能力要求较高。文中利用关键字驱动的思想,将测试脚本中变化的逻辑和数据部分从测试脚本中分离出来,放到单独的测试数据表格中。实现了一种对 Java 的应用程序编程接口进行自动测试的框架 JAPITest。该框架实现了 API 的自动提取、关键字的自动生成、测试用例管理和执行环节的自动化,具有测试脚本的可重用性高、测试逻辑清晰、测试用例维护方便等特点。

关键词:关键字驱动;API 测试;测试自动化;Java 测试;框架
中图分类号:TP391 **文献标识码**:A **文章编号**:1673-629X(2011)08-0042-04

A Keyword-Driven Automated Testing Framework for Java API

GAO Qing-juan, CHEN Li-chao, ZHANG Ying-jun, PAN Li-hu, XIE Bin-hong
(School of Computer Science and Technology, Taiyuan University of Science and Technology, Taiyuan 030024, China)

Abstract: The traditional API testing need to write code to drive measured API, compile the driver program, then execute the driver, the last is to do analysis on the test results. The process is trivial and tester requires higher code ability. In order to solve the problem, this paper proposes an automatic test framework(JAPITest). Based on the keyword-driven automation theory, the framework separate test logic and data from test scripts. An example illustrated superiorities of the framework. The framework reduced the scale of test scripts. The test logic is clearer, and the test cases maintenance is more convenient.

Key words: keyword-driven; API test; test automation; Java testing; framework

0 引 言

软件测试是提高软件开发质量的一个关键环节,随着软件规模的扩大,软件设计复杂程度的不断提高,软件开发中出现错误或缺陷的机会也越来越多,原来的手工测试难以满足需要。实现软件测试的自动化,提高软件测试的速度和效率就成为当前的研究热点。

软件提供的接口类型一般有两种:图形化用户界面(GUI)和应用程序编程接口(API)。对于图形化用户界面的自动测试目前有一些商业的工具[1,2],如 AutoRunner 等。而对应用程序编程接口的测试需要手动编写测试代码,对测试人员的专业水平要求较高,测试过程复杂[3]。目前,对应用程序编程接口测试自动化的研究主要集中在根据接口信息自动生成测试用例方面[4,5]。API 的自动测试框架有对 Java 接口进行测试的自动化框架 Junit[6],用于编写和运行可重复的测试,使用它实现自动测试时可以提高测试效率。但并不支持关键字驱动测试,测试脚本的可重用性不高。基于此,文中基于关键字驱动的思想[7,8],将测试脚本中变化的逻辑和数据从测试脚本中分离出来,放到单独的测试数据表格中,提出并实现了一种自动测试框架 JAPITest(Java Application Interface Test System)。

1 JAPITest 框架测试思想

关键字驱动测试是对数据驱动测试的一种改进形式[9]。数据驱动测试中只将测试数据从脚本中分离出来,而测试逻辑仍嵌在测试脚本中,如果测试逻辑不同的话,脚本便不能共享。而关键字驱动测试就是将测试逻辑也从脚本中剥离出来,这样就进一步增大了脚本的共享程度,使测试脚本的数量减少。

将测试用例用结构化的短文本描述是建立基于关键字的自动测试的前提,就是将构成任何应用程序的离散功能性业务事件使用关键字和相关联的参数值进行描述[10]。

收稿日期:2011-01-23;**修回日期**:2011-04-07
基金项目:山西省自然科学基金资助项目(2009011022-1)
作者简介:高青娟(1980-),女,硕士研究生,主要研究方向为智能软件、软件测试技术;陈立潮,博士,教授,硕士生导师,CCF 高级会员,主要研究方向为智能软件、人工智能。

1.1 关键字设计

该框架设计了功能关键字、系统关键字和用户自定义关键字等三种类型的关键字。每个关键字由关键字名称、关键字参数和关键字描述三部分组成。关键字名称唯一标识一个关键字,不允许重名。关键字参数是执行该关键字所必需的附加数据。比如对于"login"关键字,执行时一般还需要登录的用户名和密码两个参数。关键字描述用来描述关键字的功能和行为。

功能关键字是由被测试应用程序接口生成的关键字,和被测试程序功能一一对应。由框架自动生成。

系统关键字提供一些基本的系统功能:如导入资源包、定义变量,提供循环功能等,还有一些验证类型的关键字,如 assertEquals、assertTrue。系统关键字与被测程序无关。本框架提供的部分系统关键字见表 2。

表 2　系统关键字表

关键字名	参数 1	参数 2	描述
import	类名		导入相应的类
assertEquals	对象 1	对象 2	比较对象 1 和对象 2,相等返回真,不等返回假
assertTrue	Boolean 对象		

用户自定义关键字由一组功能关键字或系统关键字组合而成。有效的关键字框架将允许测试设计员把常用的一组关键字组合成一个业务级的关键字。业务级关键字处理完整的业务事件,而不是组成业务线程的离散功能事件。

1.2 测试用例表示

测试用例由若干测试步骤和测试验证部分组成,一个测试步骤由一个关键字名和关键字参数组成。关键字可以是功能关键字、系统关键字和用户定义关键字。测试验证部分由若干以 assert 开头的系统关键字组成。如果一个测试用例有返回值,一般对返回值进行验证;如果没有返回值,它一定会对程序产生其它一些影响,比如改变了一个对象的状态,触发了某个事件,向外部媒体写入了某些数据等,这种情况应该验证该 API 行为产生的影响是否和预期结果相一致。

2 JAPITest 框架实现

2.1 框架总体结构

JAPITest 框架主要由关键字库管理模块、测试用例管理模块、测试执行模块三个模块组成。测试框架的总体结构如图 1 所示。

2.2 框架关键模块实现

关键字库管理模块实现对关键字的添加、删除、查询等功能。它包含四个子模块,分别是 API 信息提取模块、功能关键字模块、系统关键字模块、用户自定义关键字模块。

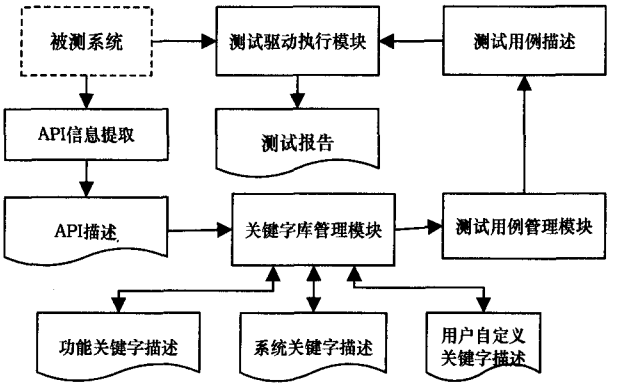

图 1　测试框架整体结构

对于测试过程中的一些中间结果,如 API 信息、关键字信息、测试用例表示等,用 XML 文件进行存储,因为 XML 具有良好的格式,并且能够自定义标签、易扩展,适合存储结构化数据[11]。

API 信息提取模块从被测软件中提取接口信息,存入 API. XML 文件中供其它模块使用。提取方法主要用到 Java 的反射机制[12]。需要提取的信息有类名、是否静态类、方法名、是否静态方法、参数个数、每个参数的类型等。

功能关键字模块提供对功能关键字的生成、查询功能。包括依据 API 信息自动生成相应的功能关键字、对功能关键字的相关信息查询。

测试用例管理模块实现对测试用例的添加、删除、修改、查询功能。

测试执行模块执行测试用例并验证结果产生测试报告。测试用例由关键字组成,所以具体就是对关键字的解释执行。

图 2 是解释执行单个功能关键字和系统关键字的程序流程图。

3 实例应用

由于篇幅所限,文中不可能用一个例子来涵盖框架的所有细节。所以下面仅以一个简单的例子说明该框架的使用过程,并与其它测试框架进行比较。被测程序如下:

```
//货币类
public class Money{
    private int Amount; //货币数量
    private String Currency; //货币符号
    //各方法的具体实现代码省略
    public Money(int amount,String currency){}
    public int getAmount(){}
    public String getCurrency(){}
    public void add(Money m){}
    public void sub(Money m){}
}
```

```
//利率类
import java.util.*;
public class Rate{
    public static void addRate(String currency1,String currency2,
double rate){}
    public static double getRate(String currency1, String
currency2){}
}
```

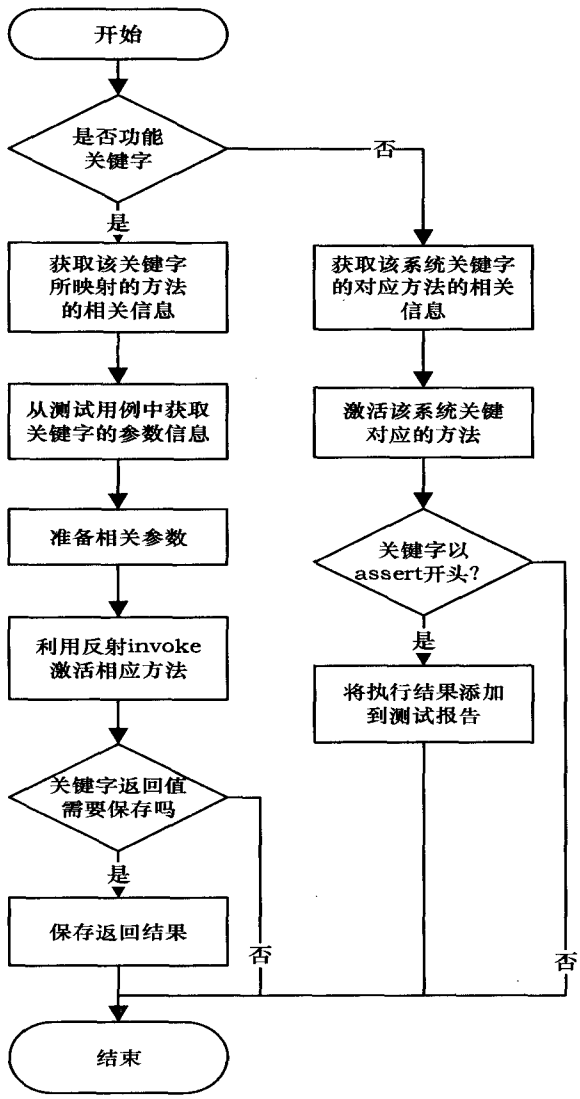

图2 关键字解释执行流程图

首先API提取模块提取API信息,关键字管理模块根据API信息自动生成功能关键字如图3所示。下面是功能关键字管理模块自动生成的部分功能关键字的XML文件。

```
<? xml version="1.0" encoding="UTF-8"? >
<keywords>
    <keyword keyword_name="createMoney0" isConstructor
="true">
        <parameterList>
            <parameter>java.lang.String</parameter>
            <parameter>double</parameter>
        </parameterList>
    </keyword>
    <keyword keyword_name="MoneydogetAmount" isCon-
structor="false" isStatic="false" method_name="getAmount" re-
turn_type="double">
        <parameterList>
            <parameter>Money</parameter>
        </parameterList>
    </keyword>
    <keyword keyword_name="Moneydoadd" isConstructor
="false" isStatic="false" method_name="add" return_type="
void">
        <parameterList>
            <parameter>Money</parameter>
            <parameter>Money</parameter>
        </parameterList>
    </keyword>
</keywords>
```

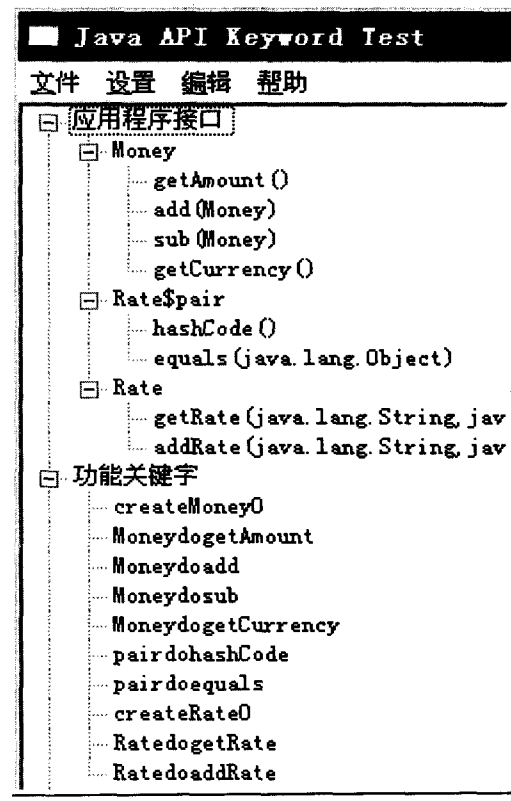

图3 接口信息和功能关键字

然后设计测试用例,这里设计了6个,为这6个测试用例编写测试脚本和对应的测试数据表。图4是测试脚本,图5是测试脚本所对应的测试数据表。

最后执行测试用例,执行结果如图6所示。可以看出有两组测试用例失败,分别是第2组和第4组测试用例,在测试步骤的第5行与预期结果不一致。

设计同样的测试用例,用Junit进行测试时,需要为add和sub功能分别编写测试脚本,对多组测试数据的组织需要在测试脚本中手动处理。

赋值对象	关键字名	参数1	参数2	参3
m1	createMoney()	RMB	V	
m2	createMoney()	USA	V	
	RatedoaddRate	USA	RMB	V
	V	m1	m2	
expected	MoneydogetAmount	m1		
	assertEquals	V	ex...	

图 4　测试脚本

变量1	变量2	变量3	变量4	变量5
10	15	6.7	add	110.5
5.4	1.5	6	add	14.4
24	-5	6.4	add	-10
10	15	6.7	sub	-90.5
25	1.5	6	sub	20
24	6	7	sub	-18

图 5　测试数据表

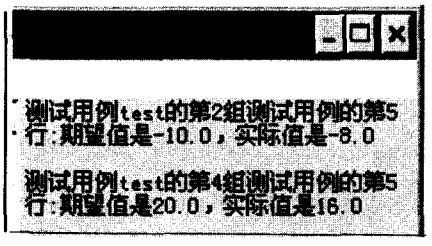

图 6　测试结果

通过比较,可以看出 JAPITest 测试框架至少有以下几点优势:

(1)有效减少了测试脚本的数量,增强了测试脚本的可重用性。以往如果测试逻辑不同必须重新编写测试脚本,在该框架中只要将脚本中关键字部分设置为变量,将具体操作放到测试数据表格中。

(2)将测试脚本的变化的逻辑和数据部分从脚本的分离出来,放到单独的测试数据表中,使测试逻辑更加清晰,测试用例维护更加方便。

(3)不用编写代码,仅需提供测试用例描述表和测试数据表。

由于每个关键字只能实现单一的功能,而编程语言可以嵌套,所以可能出现测试用例步骤描述行数比编写代码的行数多的情况。例如代码 assertEquals(2, Money. getMoney());只用一行,但在测试用例步骤描述中必须用两行:先用关键字 getMoney 得到货币数量,然后再用 assertEquals 比较。

4　结束语

为了简化 API 的测试过程,有效减少测试脚本的数量,提出并实现了一种自动测试框架 JAPITest。该框架将测试逻辑和数据从测试脚本中分离出来,提高了测试脚本的可重用性,减少了测试脚本的规模,而且使测试逻辑更加清晰,便于测试用例的维护。框架实现了对 API 的自动提取、测试用例管理和执行环节的自动化,这就使测试人员可以将精力集中在测试用例的设计上,节省了测试的工作量。下一步的工作是进一步丰富系统关键字的内容,比如提供判断分支、循环等系统关键字等。

参考文献:

[1] 隋　颖,于秀山,杨　豹. GUI 软件测试文档辅助工具的设计与实现[J]. 计算机技术与发展,2009,19(12):146-149.

[2] Cyrille A, Armin B. Advanced unit testing: How to scale up a unit test framework[C]// Proceedings of the 28th International Conference on Software Engineering, ICSE. [s. l.]:[s. n.],2006.

[3] 刘慕涛,张　磊,王　艳,等. 基于 XML 的 API 自动化测试工具设计与实现[J]. 计算机工程,2007,33(13):96-98.

[4] 赵斌飞,刘　磊. 测试脚本自动生成器的设计与实现[J]. 计算机科学,2008,35(6):276-279.

[5] 范明红,浦云明,汪志化. 等价类测试与划分研究[J]. 计算机技术与发展,2009,19(7):62-65.

[6] J. BRainsberger Scott, Stirling. Junit Recipse 中文版[M]. 北京:电子工业出版社,2006.

[7] 冯玉才,唐　艳,周　淳. 关键字驱动自动化测试的原理和实现[J]. 计算机应用, 2004,24(8):140-142.

[8] Feng, Li, Zhuang, et al. Action-driven automation test framework for Graphical User Interface (GUI) software testing [C]//Baltimore, MD, United states: 42nd Annual IEEE AUTOTESTCON Conference. [s. l.]:[s. n.],2007.

[9] Mosley D J, Posey B A. Just enough software test automation [M]. America: Prentice Hall,2002.

[10] 宋　群,朱怡安,袁文成. 关键字驱动脚本技术的应用[J]. 计算机工程与设计,2007,28(21):5091-5093.

[11] 郝晓晓,张卫丰. 基于 XML 的 SDK 自动化测试框架的设计与实现[J]. 计算机技术与发展,2010,20(4):101-104.

[12] Horstmann C S, Carnell G. Java 2 核心技术[M]. 北京:机械工业出版社,2000.

(上接第 41 页)

[s. n.],1992:300-307.

[9] Gustavson B B, Theus J. Wire-OR Logic on Transmission Lines[J]. IEEE Micro.,1983,3(3):51-55.

[10] ARINC Backplane Data Bus, ARINC Specification 659[S]. 1993.

[11] 李仁刚,秦济龙. 一种收发逻辑环回测试方法的设计与实现[J]. 科学技术与工程,2010(9):38-42.

[12] 郭跃云,杨明涛,郑　宾. 基于 DSP 的 ARINC429 总线收发系统设计[J]. 仪表技术与传感器,2010(5):56-57.

[13] 詹辉辉. 基于 CMOS 工艺的高速 CAN 总线收发器的设计与研究[D]. 长春:吉林大学,2008.

[14] 王一竹,薛海卫. 一种辐照加固的航空总线收发器电路设计[J]. 电子与封装,2009(4):25-29.