关键字驱动的自动化测试框架设计与实现

王磊

（枣庄学院 计算机科学系，山东 枣庄，277160）

摘要：为了提高软件自动化测试脚本的可复用性，本文提出了一种基于关键字驱动的自动化测试框架。框架以关键字驱动思想为核心，在设计自动化测试平台的过程中实现了测试逻辑、测试脚本和测试数据的分离，仅通过对控制文件的修改就可以实现相应测试，同时，测试脚本不关心测试用例，测试的数据和业务逻辑都集成在测试数据表格之中，测试的设计就简化为测试数据表格的设计，最大程度地减少了相互之间的影响。进一步把测试工程师从繁琐的重复性劳动中解放出来，为软件产品提供更为高效的、精准的测试，提高产品的竞争力。

关键词：关键字驱动；自动化测试；测试框架

中图分类号：TP311.52　　**文献标识码：**A

Keyword-driven test automation framework design and implementation

Wang Lei

(ZaoZhuang University, Shandong, Zaozhuang, 277160)

Abstract: In order to improve software reusability of automated test scripts, this paper brings up test automation framework based on keyword-driven. This framework whose core is the thought of keyword-driven realizes the departure of test logic, can realize testing through the revising of controlling files. Meanwhile, testing scripts don't have to concern testing examples, testing data and operation logic are stored in testing data table, and thus the design of testing can be simplified to be the design of testing data table, maximum reducing the impact of each other. Further frees the test engineers from the tedious repetitive work, providing software products more efficient, more accurate testing and thus increasing competitiveness.

Keywords: Keyword-driven; automated test; test framework

0 引言

随着软件系统的日趋复杂以及回归测试等重复性测试在整个软件生命周期中所占的重要地位，我们必须使用自动化测试技术来提高我们的测试效率。自动化测试技术能帮助软件开发人员和测试人员在更短时间内开发出更高质量的产品，通过代替频繁重复的手工测试从而节省了大量的时间和开支。

但是利用捕捉 / 回放测试工具本身无法提供高效的测试。捕捉产生的脚本对于应用的变化过于敏感，以至于测试人员要不停地修改测试脚本。这样的测试脚本不是我们想要的。我们需要的是一个易于维护的，可以应用于各种不同应用的测试模型。

1 传统自动化测试方式

使用传统自动化测试方式对测试对象各规则要点进行测试的一般流程如图 1 所示，具体如下：首先等待测试对象构建生成；得到测试对象后，判断对象该规则要点的测试是否需要新的测试数据，例如测试对象“证件号码”的测试规则要点为：必须是数字、长度为 4 个字符。判断结果有两种情况：（1）如果测试需要新的测试数据，原来录制的测试脚本将不能继续使用，必须录制新的脚本以适应当前测试数据的改动，之后再回放测试脚本以验证测试对象该规则要点；（2）如果测试不需要新的测试数据，则直接回放原测试脚本验证对象规则。测试对象该规则要点验证完毕后，继续进行下一规则要点的验证，直至测试对象各规则要点验证完毕。

可见，按照传统方式进行测试具有：比较容易使用，测试人员不需要额外的编程经验；用例生产简单，需要设计的工作很少；直观的一步一步执行方式等优点。

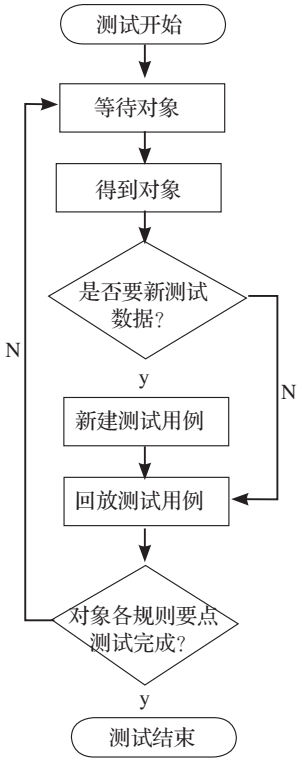

图1 传统测试流程

但是，这种简单的录制、回放过程在实际应用中存在很多问题，最常见的一个问题是测试脚本难以重用，当对程序界面进行录制时，自动化功能测试工具记录程序执行的全部过程，测试逻辑、测试对象、执行动作和测试数据包含在一个脚本中，一旦测试的执行顺序或测试对象有任何变动，都可能造成测试用例被破坏，需要手工修改已录制好的相应测试脚本，或者重新进行一次录制。尤其对于测试对象中需要多条验证的测试规则，它的改动会引起大量测试工作的返工，所有相关脚本都会受到影响，造成测试脚本的日常维护工作量急剧加大，维护这样的脚本是十分困难的。

2 关键字驱动的思想

如何解决传统测试过程中遇到的这些问题呢？

只有修改测试脚本的结构，才能从根本上解决这个问题：

2.1 界面元素名与测试工具定义对象名的分离

可以在被测程序和生成的测试脚本之间增加一个模型层，它可以将界面上的所有元素映射成对应的逻辑对象，测试针对这些逻辑对象进行，界面元素的改变只会影响映射表，而不会影响测试。

2.2 执行动作与具体实现细节的分离

把测试执行的动作和测试具体实现细节分离开来，用关键字描述测试执行动作，只说明该步测试执行什么动作而不管测试工具具体怎样执行。这样做是因为测试的实现细节通常和特定的测试执行工具有着密切的联系，比如 QTP 和 RTF。这种分离使得关键字对于实现细节不敏感，有利于测试在不同工具间的移植。

2.3 测试脚本与测试数据的分离

最后，可以把测试执行过程中所需的测试数据从脚本中提取出来，在运行时由测试控制模块从数据库中读取预先定制好的数据，这样测试脚本和测试数据可以独立维护。

采用上述关键字驱动自动化测试的思想，使执行动作、测试对象和测试数据相互独立，最大程度的减少相互之间的影响，彻底解决了使用 GUI 自动化测试工具产生的问题。

3 软件自动化测试框架

该测试框架从逻辑上自底向上分为 4 层，如图 2 所示，分别是：由测试对象和识别脚本构成的模型层，由标签库和测试数据库构成的数据层，由控制文件构成的控制层以及由用例描述原语（ASL）构成的应用层。本文以自动化测试工具 QTP 为例，搭建具体的测试框架。

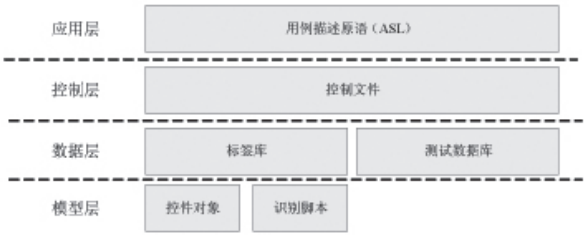

图2 软件自动化测试框架

3.1 模型层

框架最底部的模型层由测试工具识别的 GUI 控件对象以及对象的识别脚本构成，模型层负责将 GUI 控件对象和识别脚本分别组织并以友好的方式提供给数据层的标签库模块使用。

（1）对象识别脚本

在使用 QTP 录制时，QTP 自动记录 GUI 对象并生成测试脚本。该测试脚本包含了对 GUI 对象的识别、执行动作和测试数据，使得测试脚本难以重用和维护。本文在测试框架模型层中只保留对 GUI 对象的识别脚本，而把测试数据放到数据层、执行动作放到控制层管理，从而实现了相互之间的分离。

（2）GUI 控件对象

在测试框架模型层中，通过对界面元素的分析（UI 分析）并和 QTP 对象识别机制相映射来组织 GUI 控件对象。UI 分析是从被测系统界面入手，记录页面层次并把该页面中的 GUI 对象以要素的形式记录到数据库中，如图 3 所示，以达到结构化、

层次化管理对象的目的。

UI_ID	描述	模块	来源	EITEL	数量	要素1	要素2	要素3	要素4	要素5
MI671_1	现金存取选择	现金存取模块	现金存取		1	现金存取标题				
MI671_2	现金存取	现金存取模块	现金存取		10	币种映射	账号类型	账号输入	11存/99取	操作类型
MI1422_1	权证行权选择	其他委托	权证行权		1	权证行权标题				
MI1422_2	权证行权	其他委托	权证行权		9	账号类型	账号输入	权证行权	操作类型	代理证件类型
MI1422_3	权证行权查询撤销	其他委托	权证行权		1	选择记录				
MI1422_4	权证行权条件选择	其他委托	权证行权		9	账号类型	账号输入	委托方向	证券代码	客户网点
MI4200_1	单客户查询选择	查询	单客户查询		1	单客户查询				
MI4200_2	单客户查询	查询	单客户查询		4	账号类型	账号输入	居间人	资产账号	
MI4200_3	单客户查询选择	查询	单客户查询		11	客户信息	机构信息	资金	证券	基金

图3 GUI控件对象映射

3.2 数据层

数据层由标签库和测试数据库构成，标签库保存要素名和识别脚本的映射关系，测试数据库保存具体的测试数据。

（1）标签库

标签库是针对某一个应用建立的，它可以通过学习而扩展，其中包含了图形用户界面交互的完整细节，包括 GUI 界面和对象。本文通过打标的方式来对具体交互细节进行封装：首先利用 QTP 录制待测系统的执行过程，获取各测试对象的脚本；然后对录制脚本进行打标，实现要素名和测试对象的对应；最后对打标后的脚本进行分解并提取识别脚本和标签，生成标签库。例如表 1 中测试脚本中的“开户 . 客户姓名”即为识别脚本 Browser(" 系统 – 框架页面 ").Page(" 系统 – 框架面).Frame("Frame").WebEdit("textfield422") 的标签。

表1 脚本标记实例

Browser(″系统-框架页面″). Page(″系统-框架页面″). Frame(″Frame″). WebEdit(″textfield422″). Set “admin” ‘开户. 客户姓名

通过对 QTP 脚本的分析，脚本可以分解为下述范式，生成标签库程序按照相应范式对其分解、保存。

● browser.form.object.event value

该范式为 B/S 结构，其中 browser 指浏览器定位（URL），form 指 B/S 结构中的页（page），object 指用户界面的对象，event 指用户操作事件，value 指数据输入值。表 2 为 B/S 结构范式时的实例。

表2 B/S结构范式实例

Browser(″系统-框架页面″). Page(″系统-框架页面″). Frame(″Frame″). WebEdit(″textfield422″). Set “admin”

● form.object.event value

当 browser 为空时，该范式退化 C/S 结构，其中 form 指 C/S 结构中的窗口表单（form），object 指界面的对象，event 指用户操作事件，value 指数据输入值。表 3 为 C/S 结构范式时的实例。

```
Window("画图").WinObject("Afx:1000000:8").Click 107, 72
```

表3 C/S结构范式实例

● object.event value

当 form 取特殊值时，该范式退化为字符仿真终端。在该范式中，关键字对应于 event，输入参数对应于 value。

本文设计的标签库以标准 XML 的形式存储，将要素名与 QTP 识别的图形界面和对象进行映射，以供 ASL 脚本解析时使用。标签库利用 XML 元素层次关系的表达，清楚定义了业务要素所属的场景。一个业务系统生成一个标签库，不同系统的标签库不能混用。标签库实例如表 4 所示。

表4 标签库实例

```
<?xml version="1.0" encoding="gb2312" ?>
<root>
  <browser name="系统-框架页面"value="Browser("系统-框架页面")">
      <window name="开户" value="Page("系统-框架页面").Frame("Frame")">
          <object name="客户姓名">WebEdit("textfield422")</object>
          <object name="客户号">WebEdit("textfield423")</object>
          <object name="证件号码">WebEdit("textfield42322")</object>
          <object name="客户账号">WebEdit("textfield4232")</object>
          <object name="生日">WebEdit("textfield4232223")</object>
          <object name="查询">WebButton("查询")</object>
      </window>
      <window name="附属卡授权" value="Page("系统-框架页面").Frame("Frame_4")">
          <object name="授权种类">WebRadioGroup("radiobutton1")</object>
          <object name="单位结算账号">WebEdit("224")</object>
          <object name="授权方式">WebRadioGroup("radiobutton2")</object>
          <object name="授权金额">WebEdit("222")</object>
          <object name="提交">WebButton("提交")</object>
          <object name="账户列表">WebCheckBox("accountItem")</object>
      </window>
  </browser>
</root>
```

（2）测试数据库

测试数据库提供具体的测试数据，其中测试数据与 UI 分析得到的测试对象相对应，每个测试对象根据系统需要，有不同的业务规则，如果业务规则比较复杂，则把业务规则分解成最小化的规则，也即测试点。针对每一个测试点，测试人员至少要准备 2 条测试数据，分别为正例和反例。正例是指符合该测试点测试数据；反例是指不符合该测试点的测试数据。执行测试时必须保证正例能通过而反例不能通过，否则就说明该处存在 BUG，测试数据库实例如图 4 所示。

3.3 控制层

控制层为自动化测试框架提供了一个控制入口，测试工程师利用控制层实现具体的测试意图：测试工程师可以对测试的执行顺序根据需要进行调整，使得测试不必按录制顺序回放；可以通过关键字设置 GUI 对象的执行动作，实现各种操作；可以通过要素名匹配测试数据。控制文件如图 5 所示。

图4 测试数据库实例

p1	p2	p3	账号类型	客户账号	客户姓名	余额下限	余额上限	贷方累计额
测试要素：账号类型是否为必输项。	反例	预期：交易报错	null	0105014040000325	张三	12	12222222	12
测试要素：客户帐号是否为必输项。	反例	预期：交易报错	7	null	张三	12	12222222	12
测试要素：0552_操作标志是否为必输项。	反例	预期：交易报错	7	0105014040000325	张三	12	12222222	12
测试要素：客户姓名是否为必输项。	反例	预期：交易报错	7	0105014040000325	null	12	12222222	12
测试要素：余额下限是否为必输项。	反例	预期：交易报错	7	0105014040000325	张三	null	12222222	12
测试要素：余额上限是否为必输项。	反例	预期：交易报错	7	0105014040000325	张三	12	null	12
测试要素：单笔发生额是否为必输项。	反例	预期：交易报错	7	0105014040000325	张三	12	12222222	12
测试要素：贷方累计额是否为必输项。	反例	预期：交易报错	7	0105014040000325	张三	12	12222222	null
测试要素：借方累计额是否为必输项。	反例	预期：交易报错	7	0105014040000325	张三	12	12222222	12
测试要素：付费帐号是否为必输项。	反例	预期：交易报错	7	0105014040000325	张三	12	12222222	12

其中“说明”部分为生成测试报告做准备，“开始”部分设置具体测试逻辑；本例中“key”为动作关键字，框架识别该关键字并生成具体测试工具的脚本实现 UI 对象的输入，还可以设置其他关键字，比如“click”为单击，“select”为选择下拉框，“mywait”为等待一段时间等；<MZ671_2 账号输

入 > 为“MZ671_2”页面下的“账号输入”对象，程序通过标签库调用识别脚本和对象进行匹配，确定特定测试对象；“$ 账号 $”为测试数据标志，通过测试数据库的匹配找到确定的测试数据。

```
## 说明 ##
doc_casename==MZ671.简单委托.证券买入交易.网格1
doc_test_point==资产账户-资金存入，正常条件
doc_test_point==网格分析
doc_test_point==业务级
doc_test_point==正例
## 开始 ##
key==<MZ671_1.现金存取标题>==$现金存取$==现金存取标题==
mywait==1==
select==<MZ671_2.币种类别>==$币种$==币种类别==
key==<MZ671_2.账号输入>==$账号$==账号输入==
key==<MZ671_2.11存/99取>==$存取$==11存/99取==
select==<MZ671_2.操作类型>==$操作类型$==操作类型==
key==<MZ671_2.发生金额>==$金额$==发生金额==
click==<MZ671_2.确定>==$确定$==
```

图5 控制文件实例

3.4 应用层

应用层通过用例描述原语（ASL）具体实现测试的执行：ASL 首先获取测试案例的某一控制文件，得到该测试逻辑并解析出测试数据标志和标签；然后通过要素标签在标签库中查找相应脚本，匹配测试数据库中测试数据，组合生成测试工具 QTP 可以执行的脚本；最后调用测试工具 QTP 实施自动化测试。

4 基于关键字驱动技术的测试框架实例

本文以 1 个应用来分析下如何在实际中实施关键字驱动的自动化测试。在某银行开户流程的测试中，要求对所有对象进行验证，以保证软件质量。假设每个测试对象仅需要测试 1 个正例和 1 个反例，n 个对象的测试组合就是 2n 次，利用本测试框架，只在测试开始时建立框架体系，对控件对象识别、打标并生成标签库，对测试规则分析设计出测试用例。通过这种方式使测试逻辑、测试脚本和测试数据相互脱离，在回归测试中仅通过对控制文件的修改就可以对相同功能、不同数据的用例进行测试，同时，测试脚本不关心测试用例，测试的数据和业务逻辑都集成在测试数据表格之中，测试的设计就简化为测试数据表格的设计，程序运行实例如图 6 所示。

图 6 测试框架工具运行实例

由上述测试实例可以看出，应用关键字驱动技术进行自动化测试，可以把测试工程师从繁琐的重复性劳动中解放出来，也为软件产品提供更为高效的、更为精准的测试，提高产品的竞争力。

参考文献

[1] 王磊，罗省贤 . 业务流程路径覆盖方法的研究与实现 [J] . 电子测试 ， 2009(01):15–19, 52.

[2] 陈越 , 刘强 , 陈玉健 . 基于 GUI 的面向对象软件回归测试技术研究 [J]. 计算机应用研究 , 2006, 23(5): 49–51.

[3] 蔡维德 , 白晓颖 , 陈以农 . 浅谈深析面向服务的软件工程 [M]. 北京 : 清华大学出版社 , 2008: 3–11.

[4] Chen Y N, Tsai W T. Distributed Service–Oriented SoftwareDevelopment [M]. Dubuque: Kendall/Hunt Publishing,2008: 11 – 30, 271– 274.

[5] MENDEL JM, ROBERT I J, LIU Feilong. Interval type–2fuzzy logic systemsmade simple[J]. IEEE Transactions on Fuzzy Systems, 2006, 14(6): 808–821.

[6] 陈薇 , 孙增圻 . 二型模糊系统研究与应用 [J]. 模糊系统与数学 , 2005, 19(1): 126–135.

[7] 姚实颖 , 肖沙里 . 软件测试自动化建立可维护脚本 [J]. 计算机工程 , 2003 ,7 (11).

[8] 黄陇 , 于洪敏 . 基于 UML 的软件测试自动化研究 [J]. 计算机应用 , 2004, 7 (7).

[9] 黄茂生 . 三层脚本的自动化测试实现 [J]. 软件可靠性与测评 , 2005,12 (12).

[10] 姚实颖 , 肖沙里 , 谭霞 , 唐跃林 . 软件测试自动化中建立可维护脚本的技术 [J]. 计算机工程 , 2003,(11) .

作者简介：

王磊，枣庄学院计算机系教师，硕士，主要从事软件测试、分布式网络与并行计算研究。

E–mail：look_00@163.com

（上接 3 页）

4 结束语

综上所述，实验验证对于目标检测来说，彩色图像其本身的颜色信息是及其重要的。将彩色图像在 RGB 空间分为 3 个颜色层，在每一图层中，目标与背景的亮度信息都存在差异性，合理地利用这一特性对目标检测是一个有效的途径。检测时，分别对 3 层图像做检测运算、阈值分割，可以获得 3 个不同层面的检测结果，最后将所得 3 个结果进行融合，获得的检测效果具有更好的完整性和准确性。

参考文献

[1] R.C.Gonzalez. 数字图像处理 (MATLAB) 版 [M]. 阮秋琦等译 . 北京 : 电子工业出版社 ,2005:201–210.

[2] 张笃振 , 刘淑娥 . 三种基于空间转换的彩色图象分割算法比较研究 [J]. 电脑知识与技术 , 2006, 31(9): 74–76.

[3] P.Soille. 形态学图像分析原理与应用 [M].2 版 . 王小鹏等译 . 北京 : 清华大学出版社，2008:306–308.

[4] 王咏胜 . 基于数学形态学的灰度图像的边缘检测技术研究 [D]. 哈尔滨 : 哈尔滨工程大学 ,2007:36–37.

[5] 宋 鸣 . 基于数学形态学的图像分割及其在医学图像中的应用 [D]. 扬州 : 扬州大学 , 2005:45–47.

[6] 张志龙 . 基于遥感图像的重要目标特征提取与识别方法研究 [D]. 长沙 : 国防科学技术大学 ,2005 ; 38–40.

[7] 阮秋琦 . 数字图像处理学 [M].2 版 . 北京 : 电子工业出版社 ,2007:102–105.

[8] 崔屹 . 图像处理与分析 – 数学形态学算法及应用 [M]. 北京 : 科学出版社 ,2002 ; 246–248.

作者简介 ：

霍富功，主要从事图像处理、图像测量、计算机视觉方面的研究与学习。

E–mail:huofugong5519@163.com