一种分层结构测试脚本技术①

万 琳，廖飞雄

(装甲兵工程学院 信息工程系，北京 100072)

摘 要：测试脚本是自动化测试重要的组成部分，对不同类型的脚本技术进行了研究总结，提出且实现了一种分层结构的脚本技术，并对其进行了评价。结果表明该技术可行，达到了预期的目标。

关键词：软件质量保证；脚本；自动化测试；数据驱动；关键字驱动

A Multi-Layer Test Scrip

WAN Lin, LIAO Fei-Xiong

(Department of Information Engineering, Academy of Armored force Engineering, Beijing 100072, China)

Abstract: Test script is an important component of test automation. In this paper, different kinds of test script are researched and summarized, then a multi-layer test scrip is proposed and evaluated. The result indicates that the technology is feasible and can meet the requirement.

Key words: software quality assurance; scrip; test automation; data-driven; key-words driven

软件测试是保证软件质量的重要方法，自动化测试是其中重要的一种。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]：

(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。

近几年，自动化测试得到了广泛的应用。但国外统计数据表明：“80％的自动化测试尝试是失败的”[2]。导致失败的主要原因之一在于测试脚本[3]。

测试脚本是交互式应用或非交互式应用的测试自动化中必要的组成部分，大多数测试执行工具提供的脚本语言是非常有效的编程语言。然而脚本中的信息十分广泛，信息越多，越容易出错，因而需要大量的修改、管理和维护。由于对脚本维护量的不断增大，导致自动化测试投入回报比骤降而不得不放弃。文章在研究各种类型脚本技术的基础上，并根据自身的实践经验，提出了一种分层结构的脚本技术，旨在降低脚本的维护量，提高自动化测试的成功率。

1 相关技术

目前现有的测试脚本技术大致可分为：线性结构脚本、共享脚本、数据驱动脚本、关键字驱动脚本等技术[4]。这些技术并不是相互排斥的，事实恰好相反，它们是相辅相成的。每种脚本技术在支持脚本完成测试用例的时间和开销上都有各自的长处和短处。

1.1 线性结构脚本技术

线性脚本是录制手工执行的测试事例得到的脚本。这种脚本包括所有的击键、 功能键、 箭头控制测试软件的控制键及输入数据的数字键。将线性脚本结构化，即就是线性结构脚本，其中含有控制脚本执行的指令，这些指令为控制结构或调用结构。结构化的脚本中，可以使用循环语句来执行许多重复的操作；使用调用其它脚本的功能提高脚本模块化程度。

1.2 共享脚本技术

共享脚本是指脚本被多个测试用例使用。这种技术的思路是产生一个执行某种任务的脚本，而不同的测试要重复这个任务，当要执行这个任务时只需在每

① 收稿时间:2010-11-04;收到修改稿时间:2010-11-30

个测试用例的适当地方调用这个脚本。共享脚本技术的使用朝建立迅速修改软件的自动化测试迈进了一步 。它适合于小型系统或大型系统应用中的一小部分测试。但是由于每个测试都需要一个特定的脚本，因此维护的成本仍然较高。

1.3 数据驱动脚本技术

数据驱动脚本技术将测试输入存储在独立的数据文件中，而不是存储在脚本中，脚本中只存放控制信息[5]。执行测试时，从数据文件中而不是直接从脚本中读取测试输入，这样带来的好处是一个脚本可以运行不同的测试。数据文件的格式易于处理：可以包含一些脚本运行时可以忽略的注释，但使得数据文件更易于理解因而容易维护；可以方便地选择测试数据的格式和形式，使用简单的格式来说明测试输入，以便将更多的精力放在自动测试和维护测试上。但是数据驱动脚本技术也存在一定的缺陷：编写脚本需要专业支持，即需要具有一定编程背景知识的人员来编写；初始建立需在花费一定的时间，开销较大。所以，数据驱动脚本可用于较大系统的测试，而不适用于小型系统的测试。

1.4 关键字驱动脚本技术

关键字驱动的测试脚本为测试数据关键字和测试逻辑关键字的有机组合，对于关键字的解释则放在相应的支持脚本中[6]。一个数据关键字代表一个测试脚本中的测试数据。在脚本运行时用真实的测试数据对数据关键字进行替换。逻辑关键字的定义与测试活动的结构密不可分，关联了测试的步骤。关键字驱动的测试脚本可以分为三层结构，分别描述测试的目标、测试用例、及测试的逻辑流程(如图 1)。

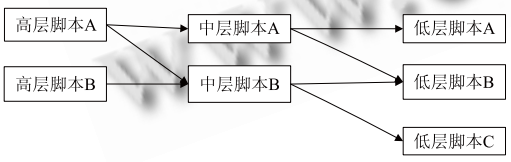

图 1　关键字脚本的组合关系

低层脚本描述了每个测试步骤的具体细节，中层脚本负责将低层脚本组装起来，执行特定的任务，高层脚本组装中层脚本形成测试循环，控制测试的执行。关键字驱动技术实际上是比较复杂的数据驱动技术的逻辑扩展，实现了脚本，数据和业务的分离，提高了可复用性和健壮性[7]。

2　一种分层结构测试脚本技术的实现

经过长期的实践总结，综合当前一些脚本技术的特点，提出了一种分层结构的脚本技术，其设计的核心思想是：

1) 脚本采用分层结构，实现测试描述与具体实现细节的分离。

2) 使用变量名从数据表中载入测试数据，实现脚本与数据的分离。

3) 用逻辑名抽象软件内部对象和操作，提高脚本的可读性。

4) 用关系表和 XML 文件结构化脚本，降低脚本的设计难度。

5) 共享支持库和底层脚本，减少脚本的数量。

2.1 脚本的组成原理

把测试脚本从高到低分为用例层、导航层、操作步骤层和应用映射层(如图 2)。为实现具体的操作细节，设立了相应的支持库（控件对象库和动作库）。

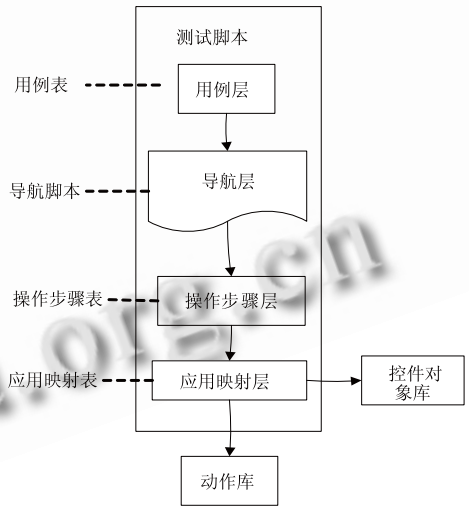

图 2　测试脚本的组成图

2.1.1 用例层

用例层是测试脚本的最高层次，通过关系表对所有的测试用例进行宏观上的描述，称之为用例表。在用例表中存放用例名及用例综述信息。用例层只关注于测试用例对功能点的覆盖情况，而不必考虑其具体的操作过程，从而实现了用例描述与具体实现细节的分离。这样做的另一个好处是测试用例设计人员通过用例层在设计用例时可以很快的根据测试需求说明列出所有的用例，在对用例进行维护时可以很快的检查

测试用例对功能点的覆盖率及缺失的用例。

2.1.2 导航层

导航层实现从用例层到操作步骤层的映射，只关注于用例的操作过程、测试输入数据的存放位置，不必考虑用例的设计和操作实现细节。其脚本存储在 XML 文件中。XML 是一种开放的定义严格的可扩展标记元语言，使用 XML 作为测试脚本，可以降低测试脚本与测试工具的耦合度和脚本的编写难度，并提高测试系统的灵活性和可扩展性[8]。

2.1.3 操作步骤层

操作步骤层通过对被测软件的界面进行分析，结合软件的使用说明等开发文档，抽取驱动被测程序执行测试时的所有操作。抽取的操作在存放在关系表中，称其为步骤表，表中包含了操作步骤名及对操作的描述信息。操作步骤名是对操作抽象出的逻辑关键字。操作描述信息的目的是对具体操作的解释，方便其他人理解，便于以后的维护和复用。操作步骤层只关注操作本身，不必考虑上层的业务及下层操作的实现。

2.1.4 应用映射层

应用映射层完成每一个操作步骤到具体实现细节的映射，只考虑每个操作步骤实现细节。应用映射表将操作步骤名同控件逻辑名、动作名及参数关联起来。系统再通过控件逻辑名映射到被测软件内部真实的控件对象；通过动作名映射到相关控件对象的动作函数；参数是待测数据的变量名，为操作载入测试数据。

2.1.4 支持库

图形用户界面 GUI(Graphical User Interface)是大部分应用软件与用户交互的重要手段[9]。自动化测试工具必须能够识别 GUI 控件对象并实现其动作才能完成测试的操作，因此设立了支持库。支持库包括控件对象库和动作库：在对象库中，用外部逻辑名标识每个控件对象，建立了被测程序内部真实界面控件对象与外部逻辑名之间的映射，让测试脚本具有更好的灵活性、可读性和更大的修改空间[10]；在动作库中，通过动作名与各种控件进行操作(如输入文本、鼠标单击、勾选、选择文本等)的动作函数的关联，实现动作关键字与操作具体实现细节之间的映射。测试系统最终要靠支持库来完成具体的操作细节。

这种分层结构的脚本，把用例的描述到实现分成了多个层次，每个层次只完成相对独立且特定的任务，通过相邻层次脚本的映射关系来实现脚本的关联组合。

2.2 脚本的设计

脚本的设计是参考测试用例的设计过程进行的，两者可以同时进行，方法具体如下：

1) 设计用例表。根据被测软件需求说明、软件应用领域的相关知识等确定测试的内容，列出将要执行的所有用例，并对测试用例进行宏观描述。

用例表={(用例名 $_i$，用例综述 $_i$)|i=1,2…}

2) 设计步骤表。根据被测软件的界面、需求说明、操作说明等文档抽象出测试中将要采取的操作。

步骤表={(操作名 i，操作描述 i)| i=1,2…}

3) 设计导航脚本，并填写数据表。测试人员根据测试用例的执行过程为其映射具体的操作，并设计测试数据，填入数据表中，并定义载入数据的变量名。

导航脚本={测试用例 i|i= 1,2…}，而测试用例={数据位置，步骤 1，步骤 2?，步骤 n}

测试数据={数据表 1，数据表 2…数据表 n}

数据表={var1,var2 … ,varn}，而 var={value1,value2…valuem}

4) 设计应用映射表根据被测程序的界面为操作关联控件对象、动作和参数(变量名)。

应用映射表={(操作名 i,控件对象名 i,动作名 i,参数 i)|i=1,2, …}

在编写测试脚本时应根据任务需求、被测软件的实际情况对人员进行合理的分工。由于各层脚本有相对的独立性，因此在进行命名约定后各个脚本可以同时进行，以便节约时间，提高效率。

2.3 脚本的运行框架

混合驱动的自动化测试框架综合了各种自动化测试框架的特点，取长补短，具有较好的灵活性和较强的通用性，是当前应用最广的自动化测试框架[6, 11]。经过研究和总结，将分层结构脚本应用在一种混合驱动的自动化测试中，并成功的实现运行(如图 3)，其基本原理是：

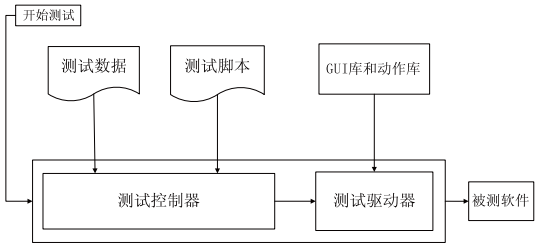

图 3 测试框架图

自动化测试启动后，测试控制器(也是脚本解析器)读取用例表，通过导航脚本找到测试数据存放位置并映射到操作步骤表，为测试用例分配操作，再通过应用映射表为每个操作分配控件对象、相应的动作及变量名形成操作序列，为变量载入数据，之后将其传递给测试驱动器，在动作库和控件对象库的支持下实现操作细节。

具体分以下几个步骤

1) 制定测试计划。根据被测软件的实际情况、运行环境，按照测试策略等进行人员分工，制定相应的测试计划。

2) 控件抽象和命名约定。综合被测软件的 GUI 界面和需求说明进行分析和抽象，约定测试用例名、操作名、控件名、动作名等各种逻辑关键字的命名方式和规则。

3) 设计测试用例和脚本。根据前面的结果，按照一定的测试用例生成规则设计测试用例和各层的脚本，并为数据表输入测试数据。

4) 扩充支持库，执行自动化测试。根据被测软件的界面录入控件和设计动作函数，并与控件逻辑名、动作逻辑名相映射，使脚本的操作能正常实现。之后启动测试工具执行测试，并生成相应的测试文档。

5) 评测测试结果。分析测试结果，完成评测文档，撰写评测报告。

3 脚本应用实例及特性分析

3.1 脚本实例

文章提出的脚本技术应用在某软件的功能测试中。该软件为某部队的大型信息管理系统，进行人员管理、装备管理、军械管理等部队的日常管理。待测软件的菜单层次平均为 4 级，最低为 2 级，最深可达 5 级。共计 4 个一级菜单，底层菜单为 127 个，每个底层菜单对应一个相对独立的功能模块。

在测试中，把同属于一个功能模块的试脚本和数据作为一个配置项，然后依次进行测试，不同的配置项之间复用测试脚本和数据。下面展示的是该软件登陆界面的测试脚本：

a.用例表(如表 1)。登陆界面上需要到用登陆和退出两个测试用例。用例表存储在 Access 数据库中，用例设计工程师可以通过数据库修改测试用例。

表 1　用例表

用例名	用例综述
LoginTest_login	能否正常登陆界面上完成登陆操作
LoginTest_quit	能否正常取消登陆，并推出登陆界面

b. 步骤表(如表 2)。登陆界面上进行测试时需要有打开程序、输入用户名和密码、登陆或取消登陆，验证是否成功登陆或是否退出程序等操作。步骤表同样存储在 Access 数据库中。

表 2　操作步骤表

操作名	操作描述
StartProgram	打开应用程序
InputName	输入登陆用户名
InputPassword	输入登陆密码
ClickLogin	单击登陆按钮
ClickQuit	单击退出按钮
VeryfyMainWindow	验证是否成功登陆进入程序主界面
VeryfyNoApp	验证是否退出程序

c.导航脚本和数据表(如图 4 和表 3)。图 4 为导航脚本，每一用例节点包括了测试数据位置节点(1 个或没有)、操作步骤节点(1 个或多个)，其中 XML 文档的根节点和其他用例节点已经省略。

```
<!-- This is the navegate scrip-->
<? xmlversion="1.0">
.....
.....
.....
<testcase> LoginTest_login
    <testdataposition> DataTable_LoginTest </testdataposition>
    <step> StartProgram </step>
    <step> InputName </step>
    <step> InputPassword </step>
    <step> ClickLogin </step>
    <step> VeryfyMainWindow </step>
</testcase>
.....
.....
<testcase> LoginTest_Quit
    <step> StartProgram </step>
    <step> ClickQuit </step>
    <step> VeryfyNoApp </step>
</testcase>
.....
.....
.....
```

图 4　导航脚本实例图

表 3　测试数据表

UserName	Password	说明
LiLi	123456	有效数据
++==	123456	无效用户名
LiLi	%$#^&	无效密码
LiLei	22222	不存在的用户名
LiLi	654321	错误的密码

表 3 为测试数据表。在登陆测试中需要对有效的用户名及密码、输入无效的用户名、输入无效的密码、输入不存在的用户名及输入错误的密码 5 个方案进行测试。

d.应用映射表（如表 4）表中包括动作操作名、控件的逻辑名、动作逻辑名和参数。第 1、6、7 条记录中的参数分别为待测程序的位置、程序主界面窗口的标题、启动的进程名或程序名(任务管理器中可以查看到)。

表 4 应用映射表

操作名	控件对象名	动作名	参数
StartProgram		startapp	C:\Program Files …
InputName	Name_edit	textinput	UserName
InputPassword	Password_edit	textinput	Password
ClickLogin	Login_button	mouseleftclick	
ClickQuit	Quit_button	mouseleftclick	
VeryfyMainWindow		findwindow	###管理系统
VeryfyNoApp		veryfynoapp	###.exe

3.2 脚本的特性分析

文章使用了一些属性作为依据来描述脚本并对脚本的特性进行分析。表 5 给出了对脚本的特性进行分析的依据。

表 5 脚本特性的分析依据

属性	好的脚本	差的脚本
结构化	结构合理、且易于理解,提供的文档清晰简洁，便于管理	组织结构混乱，修改困难,文档内容不清晰，难以管理
复用性	脚本可以用于不同的测试或测试用例	无复用，每个脚本只可用于一个用例
维护性	易于维护，软件变化只需对少量脚本进行修改	软件只要有少量变化，就需修改大量脚本，难于维护。

下面结合各个属性逐一对分层结构的测试脚本进行分析：

1) 结构化。脚本包括用例表、导航 XML 文件、操作步骤表、应用映射表，采用关系表和 XML 文件作为脚本。关系表本身就是关系数据结构，XML 文件则是一种树形结构。这两种结构均是较好的数据结构，易于理解，且修改方便，并且脚本存放在相应的数据库中，结合数据库能更方便的进行管理。

2) 维护性。①脚本存放在数据库中，通过数据库进行维护，其可维护性明显要高与直接对文档进行维护。②当软件发生变化时，只需根据变化的情况修改对应层次的脚本，修改量小。③数据与脚本进行了分离，对各自的维护更方便。

3) 复用性。脚本采用的分层结构，因此也提供了不同的复用粒度和复用方式。测试人员在不同的用例间或对不同的软件进行测试时，可以根据实际情况复用用例或用例集、复用操作、复用测试数据。

5 结语

脚本技术类似于编程技术，合理的编程开发出的软件易于维护和使用，同样合理的脚本产生的测试也同样易于维护和实现。分层结构脚本技术结合其他脚本技术的优点，实现了测试描述与细节、脚本与数据的分离，提高了脚本的可读性，降低了脚本的设计难度。经实践表明：采用关系表和 XML 文件作为脚本，存放在相应的数据库中，其结构特性增强，对其维护更加容易，对其管理更方便；依靠分层的结构提供了不用的复用粒度和复用方式，可复用性得到提高。综上所述，分层结构脚本将具有更强的生命力和发展前景。然而该脚本也增加了相应测试工具的开发难度，下一步要将工具进一步完善，同时对脚本的自动生成进行研究，提高测试的自动化程度。

参考文献

1 阚红星,龚育昌.一种基于 GUI 的测试脚本开发环境.计算机系统应用,2005,14(1):43−46.

2 Boeamer B, Atthrson PB. Software test automation-developing an in frastructure designed for Success. 2001.

3 黄茂生.三层脚本结构的自动化测试实现.软件可靠性测评,2005,12:161−164.

4 接卉,兰雨晴,骆沛.一种关键字驱动的自动化测试框架.计算机应用研究,2006,26(3):927−929.

5 Kaner C, Falk J, Nguyen HQ. Testing Computer Software (2nd edition). Wiley Computer Publishing, 1999. 143.

6 冯玉才,唐艳,周淳.关键字驱动自动化测试的原理和实现.计算机应用,2004,24(8):140−142.

7 Nagle C. Test Automation Frameworks. http://safsdev. Sourceforge.net/FRAMESDataDrivenTestAutomationFrameworks.htm,2003.

8 朱经纬.XML 技术在软件测试自动化中的应用.计算机工程,2005,31(2):94−96.

9 乔根森.韩柯,等译.软件测试.北京:机械工业出版社,2003.

10 吴恒山,姜文君.GUI 分层自动测试探索.计算机工程与应用,2007,43(3):81−83.

11 Kelly M. Choosing a test automation framework. http://www.ibm.com/developerw.orks/rational/library/591.html.