

# 关键字驱动测试的自动化框架模型与系统实现

黄威<sup>1</sup>, 聂藩<sup>2</sup>, 丁建完<sup>1</sup>

(1. 华中科技大学机械科学与工程学院 CAD 中心, 湖北 武汉 430074;

2. 苏州同元软控信息技术有限公司, 江苏 苏州 215123)

**摘要:**为了提高迭代式增量软件开发中的回归测试效率、改善测试脚本和用例的维护性和复用性, 基于关键字驱动的思想设计了层次化的测试自动化框架模型, 提出了对应的三层关键字设计方法。通过 C++ 项目-机械系统动力学 CAE 平台 MWorks 回归自动化测试框架详细的系统实现证明了模型的可行性, 合理的关键字划分隔离了测试脚本开发人员、测试用例开发人员和测试执行人员关注的问题, 有助于组织和管理测试和测试用例, 使得框架便于使用、维护和扩展。测试工具的多进程并行和异常处理进一步提高了测试效率。

**关键词:**关键字驱动; 测试自动化框架; 多进程

**中图分类号:** TP31

**文献标识码:** A

**文章编号:** 1673-629X(2012)04-0057-04

## A Model of Test Automation Framework Based on Keyword-Driven and Its System Implementation

HUANG Wei<sup>1</sup>, NIE Fan<sup>2</sup>, DING Jian-wan<sup>1</sup>

(1. CAD Center, School of Mechanical Science & Engineering, Huazhong University of

Science & Technology, Wuhan 430074, China;

2. Suzhou Tongyuan Software & Control Information Technology Co. Ltd, Suzhou 215123, China)

**Abstract:** Aiming at the improvement of the efficiency of regression testing and the maintainability and reusability of automated test scripts and cases in the iterative and incremental software development, a model of test automation framework based on keyword-driven as well as the approach for designing the keywords is brought up, and then the system implementation of an CAE platform-MWorks is presented to demonstrate the feasibility. The isolation of problems which the test script developers, the test case developers and the test executors care about by the suitable keyword classification make the framework easy to use, maintain and expand. The efficiency is further improved by multi-process parallel execution and exception handling of the test tool.

**Key words:** keyword-driven; test automation framework; multi-process

## 0 引言

回归测试是软件测试中最适合执行自动化测试的阶段。迭代增量式软件开发过程中, 每次构建完成后执行自动回归测试, 验证新版本是否纠正旧版本的缺陷, 新增功能是否导致新问题等, 保证软件质量、开发进度。关键字驱动<sup>[1]</sup>是广泛使用的一种较成熟的自动化测试技术<sup>[2]</sup>, 是对数据驱动自动化测试的有效改进和补充。关键字驱动的测试自动化框架<sup>[3-6]</sup>将具体测试、测试工具、应用程序本身的变化隔离开来, 实现了脚本、数据和业务的分离, 具有以下优点:

(1) 不同的关键字序列支持测试多样化, 覆盖更全面。

(2) 测试用例和测试脚本的分离使得测试易于维护。

(3) 测试人员只需设计基于关键字的测试用例而不再接触脚本, 使得自动化测试门槛降低。

(4) 工具的开发是一次性的, 但可复用、可扩展和可移植, 投入产出高。

文中首先提出层次化的测试自动化框架模型架构及对应的三层关键字设计方法, 然后以 C++ 项目——机械系统动力学 CAE 平台 MWorks 的自动化回归测试为例, 详细论述了该框架模型的系统实现。

## 1 自动化测试框架模型

层次化的自动化测试框架<sup>[7]</sup>模型由测试脚本库、

收稿日期: 2011-08-30; 修回日期: 2011-12-02

基金项目: 国家高技术研究发展计划(863 计划) 重点项目(2009AA044501)

作者简介: 黄威(1987-), 男, 湖北武汉人, 硕士研究生, 研究领域为多领域建模与仿真。

测试用例库、测试工程库、测试工具和缺陷管理库组成。每一种应用软件对应一个测试脚本库和测试用例库,测试工程库组织和管理测试子任务,由测试脚本集和测试用例集有机组合而成。是测试工具的输入,在回归测试中,每一个测试工程是可以重复执行的。测试工具执行测试工程,生成测试报告,包括当前测试运行日志、测试比较报告。测试完成后由测试执行人员添加测试结果到缺陷管理库。测试自动化框架模型架构如图1所示。

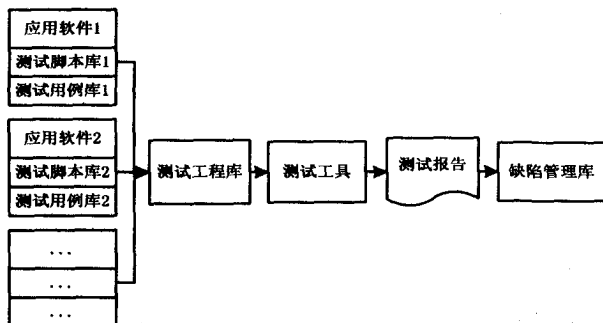


图1 测试自动化框架模型

## 2 关键字设计方法

在关键字驱动设计<sup>[8,9]</sup>中,关键字贯彻始终,设计关键字的目的是为了更好地执行测试任务。在测试自动化框架模型中,根据不同人员的关注内容从高到低将关键字设计为三层:

(1)核心关键字:面向软件用户和测试执行人员,对应软件工作流程的核心步骤,用来组织和管理测试工程。

(2)中间关键字:面向脚本开发人员,对应核心步骤的主要功能模块,用来组织和管理测试脚本库。

(3)底层关键字:面向测试开发人员,对应主要功能模块的单元算法,用来组织和管理测试用例。

## 3 系统实现

本节以C++项目——Modelica多领域物理统一建模与仿真平台MWorks的回归测试为例,详细论述了该框架模型的系统实现。

Modelica模型求解流程如图2所示:



图2 Modelica模型求解流程

编译包括检查和翻译两部分,将复杂物理的模型平坦化,铲平模型的层次结构,将模型转化为一组平坦的方程、常量、参数和变量。

分析优化阶段检查方程是否相容,若不相容则找出问题根源,以便用户校正模型;若相容则进行简化减少方程个数,为仿真求解做准备。

仿真求解时,根据方程的参数依赖关系,结合数值求解包提供的函数,形成模型的求解流程和控制策略,并生成C代码求解器,通过编译运行C程序实现模型求解。

基于以上关键字设计方法,分解模型求解流程,MWorks的测试关键字表如图3所示:

核心关键字	编译			分析		求解								
中间关键字	检查		翻译	分析	优化	求解设置		求解结果						
底层关键字	类型匹配	单位检查	函数调用	组件引用	指标约简	匹配计算	函数内联	符号简化	求解算法	求解步长	求解时间	结果分析	结果转换	结果比较

图3 MWorks测试关键字设计表

### 3.1 测试用例库

测试设计人员根据底层关键字设计测试用例,也就是Modelica源文件.mo。测试设计人员完全不需要了解Python脚本语言,将更多的精力投入到Modelica语言的研究上。测试库结构如图4所示:

测试用例库		
单位检查	函数调用	...
Connect_DeduceUnit.mo	Function_ArgulsifExpr.mo	...
Equation_DeduceUnit.mo	Function_ArguWithPre.mo	
Function_DeduceUnit.mo	Function_ArgulsCompisite.mo	
...	...	

图4 测试用例库

### 3.2 测试脚本库

开发测试脚本之前,需要软件研发人员开发Python接口。将C++接口映射为Python接口。MWorks通过CoreDiver提供Python接口,在脚本库中以脚本文件motool.py提供,其中包含所有的测试接口,形式如下:

```
import CoreDiver
...
def CheckMoModel(modelname):...
def AnalyzeMoModel(modelname):...
def CompileCModel(vcbuild, mworks, mode):...
def SimulateModel(model, setting, result):...
def CompareModelResult(analyser, result, setting):...
```

测试脚本开发人员只需了解Python脚本语言而不必了解C++。在motool.py的基础上,测试脚本开发人员根据中间关键字开发检查、翻译、分析等对应的测试脚本,求解结果测试脚本simulate.py内容为:

```
import motool
...
#编译、分析优化、求解
motool.CheckMoModel(modelname)
motool.CompileMoModel(modelname)
motool.CompileCModel(simulator, "model.c")
```

motool.SimulateModel("MWSolver.exe", setting, result)

根据中间关键字开发测试脚本库,减轻了脚本开发成本,使得测试脚本库非常简洁,便于管理和维护。

### 3.3 测试工程库

测试工程库组织和管理工作,由测试脚本集和测试用例集有机组合而成。是测试工具的输入,在回归测试中,每一

个测试工程是可以重复执行的。

每一个核心关键字对应多个中间关键字,每一个中间关键字对应一个测试脚本和多个底层关键字,一个底层关键字对应一个测试用例集。测试工程采用 XML 的格式来组织和管理这种层次化关系<sup>[10-12]</sup>。每个测试工程包含多个测试子任务,单位检查就是一个测试子任务,UnitChecking.xml 内容为:

```
<cases Name="UnitChecking", Testing="$ (py-
file) ...">
```

```
<casemoname="Connect_DeduceUnit"/>
```

```
<case moname="Equation_DeduceUnit"/>
```

```
<case moname="Function_DeduceUnit"/>
```

```
...
```

```
</cases>
```

测试工程如 Compile.xml 内容为:

```
<project Name="Compile" config="...">
```

```
<subtask Name="UnitChecking.xml"/>
```

```
<subtask Name="Function.xml"/>
```

```
<subtask Name="OuterInner.xml"/>
```

```
...
```

```
</project>
```

测试工程中可以快速添加和删除测试子任务,而测试子任务由测试编写人员维护。测试工程的引入隔离了测试活动和测试脚本、测试用例,测试执行人员不需要关心具体的测试脚本和测试用例,这是非程序员也可以完成的工作。

### 3.4 测试工具

测试工具自动执行测试工程,输出运行日志和测

试报告。测试工具是整个测试框架的核心。文中测试工具基于 Python2.5 设计和实现,解引用得到测试命令行;多进程并行执行测试用例;自动捕捉处理测试运行时崩溃,成本低而高效。测试工具运行界面如图 5 所示:

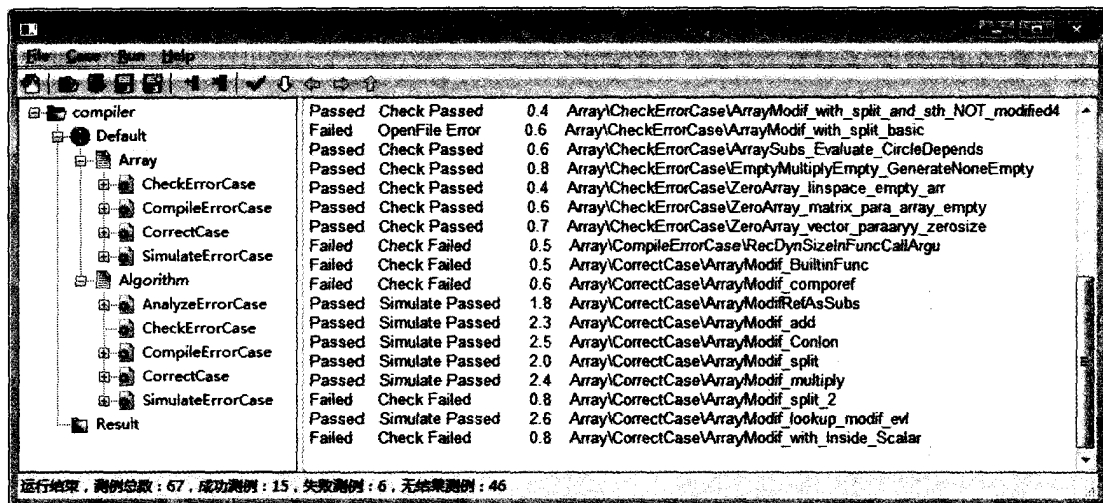


图5 测试工具示意图

#### 3.4.1 关键字和解引用

测试工具中也有一个关键字 Testing, 加载测试工程后, 解析导入相应的测试子任务, 关键字 Testing 对应的字符串即为测试工具执行的命令语句。如测试子任务 UnitChecking 对应的 Testing 形式为: Testing = "\$ (pyfile) -d \$ (coredriver) -l \$ (libs) -m \$ (moname) -t \$ (tempdir)"

由于在整个测试工程中, 所有测试的 coredriver 测试属性都是相同的; 在同一个测试子任务中, 测试脚本文件 pyfile、临时结果存放文件夹 tempdir、依赖库 libs 等都是相同的, 此处同样采用关键字的思想, 以键-值对来描述共同属性, 从而不必对每一个测试例都维护一条测试命令。便于快速的部署和修改, 如测试例 UnitChecking/ DeduceUnit.mo 解引用后的形式为:

```
Testing = 'check.py -d "... \MWorks 2.5 \Bin" -l
"Modelica.bmf" -m "DeduceUnit.mo" -t "UnitCheck-
ing\tempdir"'
```

测试工具运行测试例时, 只需要进行一次解引用, 建立测试例命令行列表, 然后启动进程顺序运行命令行自动测试。

#### 3.4.2 多任务并行执行

尽可能早地发现软件缺陷有利于保证开发进度和控制开发成本, 软件构建的周期缩短和频率加快要求自动化回归测试的效率进一步提升。但在一定的硬件和人员配备下, 大多数情况下不能并行测试; 随着 CPU 主频的升高和多核台式工作机的普及, 利用多线程或多进程来实现多任务并行执行, 可以充分地利用计算

资源, 缩减测试时间, 提高测试效率。

测试用例往往是独立的, 测例之间不需要进行通信。相对于多线程, 多进程能够满足要求, 而且更易于实现。

本测试框架中测试工具采用 Python2.5 实现, 函数 MultiRun() 通过 Python 生成器提供的 yield 关键字输出函数的中间结果, 通过管道获取进程的输出信息, 实现多进程并行。算法流程如图 6 所示:

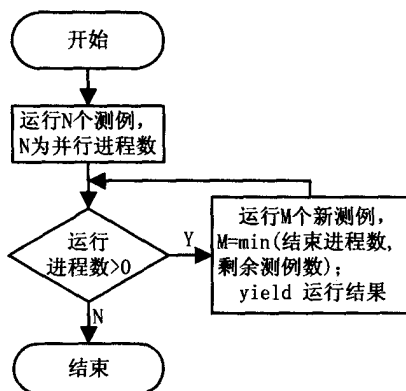


图 6 多进程运行流程图

在主程序中设置一个定时器, 定时调用 MultiRun(), 获取 yield 生成的中间结果, 输出到日志或显示到窗口。定时器定时长度取决于测试实际结果, 定时过长会造成进程空闲, 而过短则损耗资源。并行进程数也取决于硬件配置和测例占用资源比率, 因为同一时间内存资源是有限的。

本测试工具在 MWorks 回归测试中, 测试机配置为 Intel(R) Core(TM)2 Q9400 2.66GHz 4 核, 2G 内存, 并行进程数为 4 个, 定时为 100ms 时, 将 5000 个测例的运行时间由 5 个小时缩短到 1.5 个小时。

### 3.4.3 异常处理

测试工具自动化执行时, 如果不及处理由于某些运行测例导致的被测应用程序崩溃, 会中断单任务顺序测试的流水线作业; 多任务并行测试时, 崩溃测例将占用线程或进程, 降低测试效率, 当崩溃测例数超过并行测例数上限时, 测试作业将中断。因此, 在进程并发执行时, 及时处理异常显得尤为重要, 关系到自动测试能否稳定快速执行。

计时是解决该问题的一个有效方式: 监测每一个测例的运行时间, 如超过计时上限, 强制终止该线程或进程。另一种方式是通过程序模拟鼠标点击关闭窗口, 的动作关闭应用程序弹出的崩溃窗口。计时方法需要设置来源于多次实际测试结果的合理计时上限  $T_{MAX}$ ,  $T_{MAX}$  必须大于单个测例运行时间的最大值, 效率较低, 而且强制终止线程和进程实现较复杂。本框架中的测试工具采用后一种方式, 在主程序中定时调用 MultiRun() 之前, 插入检查并关闭崩溃弹出窗口

的处理步骤, 具体过程非常简单: 首先枚举桌面窗口, 检查是否存在崩溃弹出窗口, 例如 "python.exe"、"Visual Studio Just-In-Time Debugger"、"Microsoft Visual C++ Debug Library" 等, 然后获取相应的按钮控件 ID 号, 发送鼠标点击的消息, 结束崩溃测例的进程。

### 3.5 测试报告和缺陷管理库

测试工具输出运行日志和测试报告, 主要内容包括测例运行结果、运行时间、成功测例数、失败测例数和崩溃测例数。这只是测试报告的一部分; 通过新旧版本测试日志的比较进一步得到进步、衰退测例, 由测试执行人员录入缺陷管理库, 然后根据对应的测例关键字转给相关研发人员调试和修改。

## 4 结束语

根据以上分析, 首先, 关键字是开发和管理测试脚本、测试用例, 执行测试活动的主要依据。关键字的划分决定了测试自动化框架的成败。其次, 层次化模型便于分离测试、测试工具和应用软件, 使得不同身份的人可以专注于自己的领域, 节省时间和资源。测试工程采用结构化语言 XML 有机组合测试脚本和测试用例, 方便组织和管理测例库, 同时也是测试工具的输入。测试工具是测试框架的核心部分, 关键字非常适合建立索引表, 使得工具具有可扩展性; 多进程提升效率(文中尝试过采用多线程实现, 但未取得满意效果, 有待研究), 而异常处理保证运行的稳定性。最后, 值得注意的是: 基于关键字的测试自动化框架是较为困难和耗时的一种数据驱动实现方式, 维护和管理需要良好的数据库和版本管理工具支持。

### 参考文献:

- [1] Laukkanen P. Data-driven and Keyword-driven Test Automation Frameworks[D]. Helsinki: Helsinki University of Technology, 2006.
- [2] 董娜娜, 詹惠琴. 软件自动化测试技术应用研究[J]. 电子测试, 2010, 11(11): 48-50.
- [3] Tang Jingfan, Cao Xiaohua. Towards Adaptive Framework of Keyword Driven Automation Testing[C]//Proceedings of the IEEE International Conference on Automation and Logistics. Qingdao, China: [s. n.], 2008: 1631-1635.
- [4] Takala T, Maunumaa M, Katara M. An Adapter Framework for Keyword-driven Testing[C]//QSIC '09 Proceedings of the 2009 Ninth International Conference on Quality Software. Washington DC: IEEE Computer Society, 2009: 201-210.
- [5] 王 君, 朱美正, 李 欣. 关键字驱动测试框架的研究和实现[J]. 计算机工程及设计, 2010, 31(10): 2246-2248.
- [6] 李 玮. 软件自动化测试混合框架的研究和实现[D]. 北

(下转第 64 页)



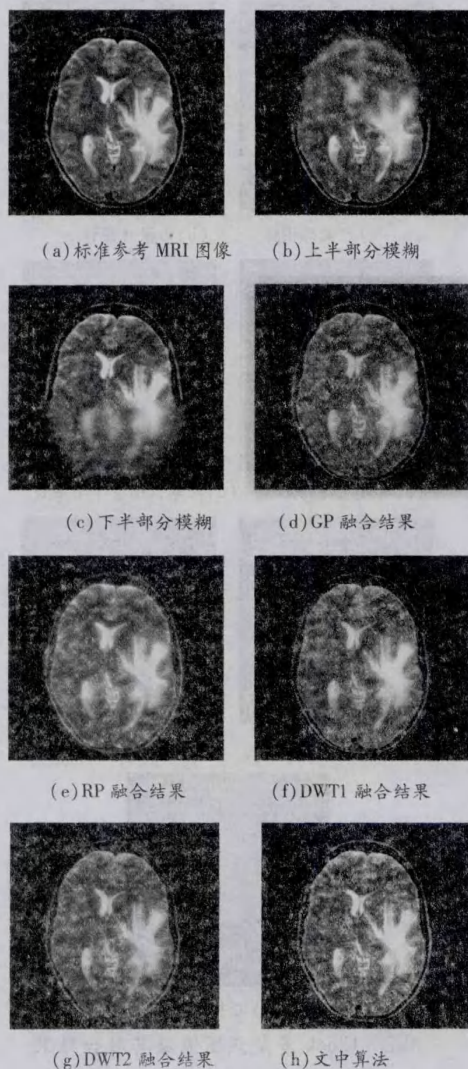


图3 不同模糊MRI图像及其融合结果

表2 不同模糊MRI图像各融合算法性能比较

融合方法	PSNR	RMSE	MI	$Q^{AB/F}$
GP	22.8053	18.4619	4.4789	0.7940
RP	24.4058	15.3550	5.1826	0.7674
DWT1	26.8162	11.6340	4.8899	0.7359
DWT2	27.6851	11.6319	4.8732	0.7352
文中算法	28.1186	10.0141	5.5834	0.8515

缘信息从源图像到融合图像的转移程度。PSNR 越大, RMSE 越小, MI 越大,  $Q^{AB/F}$  越大, 融合效果越好。

(上接第60页)

京:北京交通大学,2007.

- [7] 万琳, 廖飞雄. 一种分层结构测试脚本技术[J]. 计算机系统应用, 2011, 20(7): 141-145.
- [8] 钱月琴. 关键字驱动框架中关键字划分方法研究[J]. 计算机技术与发展, 2010, 20(9): 45-51.
- [9] 接卉, 兰雨晴, 骆沛. 一种关键字驱动的自动化测试框架[J]. 计算机应用研究, 2009, 26(3): 928-929.

## 4 结束语

文中根据现有基于小波的图像融合算法, 提出一种基于 sobel 算子和局部能量的图像融合算法。选择适当的小波和参数对文中算法与传统金字塔方法和小波变换进行实验和分析, 结果表明文中算法在目测效果和客观统计数据上都取得了较好的融合效果。

在实际应用中要注意小波的选择、分解层数的选择。分解层数越高, 计算量越大, 但融合效果不一定越好。还要注意一致性检验中参数  $N_p$ ,  $I_n$  的选择。 $I_n$  越大, 一致性检验强度越大, 但融合效果未必越好。

## 参考文献:

- [1] 赵巍, 毛士艺. 多传感器图像融合技术综述[J]. 北京航空航天大学学报, 2002, 28(8): 512-518.
- [2] Huang Shuying, Park D S, Yang Yong. Medical Image Fusion via an Effective Wavelet-based Approach[J]. EURASIP Journal on Advances in Signal Processing, 2010, 2(2): 1-13.
- [3] 李晖晖, 郭雷, 鲍永生. 图像融合[M]. 北京: 电子工业出版社, 2008: 23-27.
- [4] 李勇. 基于多尺度分解的多源图像融合算法研究[D]. 长春: 吉林大学, 2010.
- [5] 范国滨, 朱瑞辉, 万敏. 基于金字塔变换的图像融合方法[J]. 计算机仿真, 2007, 24(12): 178-192.
- [6] 吕艳琼, 於时才. 一种基于小波变换的图像融合新算法[J]. 计算机应用研究, 2009, 26(1): 390-391.
- [7] 王正志, 王超, 瞿继双. 基于数据融合的遥感图像处理技术[J]. 中国图像图形学报, 2002, 7(10): 985-991.
- [8] 杨风暴, 阳方林, 郭红阳. 像素级图像融合效果的评价方法研究[J]. 测试技术学报, 2002, 16(4): 35-39.
- [9] Petrovic V S, Xydeas C S. Gradient-based multi-resolution image fusion[J]. IEEE Transactions on Image Processing, 2004, 13(2): 228-237.
- [10] Li Shutao, Kwok J T, Wang Yaonan. Combination of images with diverse focuses using the spatial frequency[J]. Information Fusion, 2001, 2(3): 169-176.
- [11] 晁锐, 张科, 李言俊. 一种基于小波变换的图像融合算法[J]. 电子学报, 2004, 32(11): 750-753.
- [12] 任国超, 师黎. 基于2V-SVM和一致性检验的医学图像融合方法[J]. 计算机工程与应用, 2010, 46(13): 199-201.
- [10] 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): 102-104.
- [11] 朱经纬. XML技术在软件测试自动化中的应用[J]. 计算机工程, 2005, 31(2): 94-96.
- [12] Bagnasco A, Chirico M, Scapolla A M, et al. XML data representation for testing automation[C]//IEEE AUTOTESTCON Proceedings. [s. l.]: IEEE Computer Society, 2002.