WEB应用测试执行引擎的设计与实现

贺 斌,曾红卫,冯振华
(上海大学 计算机工程与科学学院,上海 200072)

摘 要:基于模型的测试以系统的抽象模型为依据,自动生成可执行的测试用例,自动地执行测试并自动评价测试结果,实现测试过程的自动化。论文主要研究测试执行自动化技术,设计实现一个自动化测试执行工具。工具的核心是设计基于XML的自动化测试脚本并实现了一个测试执行引擎。脚本拥有良好的结构定义,便于编辑和扩展,由自动化测试工具自动生成。引擎使用多线程技术模拟多用户的并行使用,解决了数据驱动测试执行方式和线程并行操作时的同步问题。

关键词:自动化测试;测试脚本;测试执行引擎
中图分类号:TP311.5 **文献标识码**:A **文章编号**:1673-629X(2011)06-0028-04

Design and Implementation of Execution Engine for Web Application Testing

HE Bin, ZENG Hong-wei, FENG Zhen-hua
(School of Computer Engineering and Science, Shanghai University, Shanghai 200072, China)

Abstract: Based on the abstract model of the system, model-based testing can generate executable test cases, execute the test and evaluate the test results automatically to achieve the automated testing process. Focus on automated testing execution technology and automated test execution tools. A test script language is designed and the corresponding tool is developed. The script based on XML can be generated by automated testing tool, which has a good structure definition and is easy to edit and extend. Test execution engine uses multi-threading technology to simulate multi-user using in parallel, and solves the problems of data-driven testing and thread synchronization in parallel operation.

Key words: automated test; test script; test execution engine

1 概 述

互联网已成为人们日常生活的组成部分,Web应用也已从一般网站发展成了大型电子商务、信息发布和信息服务平台。因而对于Web应用的测试工作也成为了软件工程领域的一个研究热点[1]。

关于Web应用测试的方法研究逐渐深入,许蕾等[1]提出了用分层表示Web模型,使用"捕捉/回放"和智能Agent等技术实现了测试执行的自动化和智能化。周晓宇等[2]改进了"捕捉/回放"技术使之适应Web应用的动态特性,并采用智能Agent来实现对整个测试执行过程的监控、管理和应急处理。文献[3]设计了基于XML的自动化测试脚本语言,该脚本语言是通过捕捉浏览器"请求/响应"数据来录制的,并对Web应用实行回放来进行测试。

收稿日期:2010-11-19;**修回日期**:2011-02-20
基金项目:国家自然科学基金项目(61073050);上海市自然科学基金(09ZR1412100);上海市重点学科建设项目(J50103)
作者简介:贺 斌(1984-),男,硕士研究生,研究方向为软件测试;曾红卫,研究员,研究领域为软件工程与形式化、软件测试。

目前,尽管也出现了一些实验性的和商品化的Web应用自动化测试工具,但大多数工具是基于"捕获/回放"机制的,实际上还是手工测试的加强版。它并不能自动生成测试用例,也不能提供测试覆盖度的任何信息[2]。另外,人工记录测试场景非常耗时,测试的维护也非常困难。

基于模型的测试以明确描述系统预期行为的抽象模型为依据,自动生成可执行的测试用例,自动地执行测试并自动评价测试结果,实现测试过程的完全自动化。因此基于模型的测试技术是解决Web应用测试问题的有效途径。测试执行引擎是整个自动化测试工具的重要部分,由测试脚本和输入数据所驱动。执行引擎模拟多用户的操作行为对被测Web应用进行功能测试。经过测试序列实例化及测试数据生成等前期工作的完成,文中将给出Web应用测试自动化工具中测试执行引擎的设计及实现。

2 测试执行引擎的设计

测试引擎根据Web应用三层结构所设计,分别对

应了客户端、服务器端及数据库端[4]。

引擎使用多线程技术,实例化一组客户端来模拟多用户同时对Web应用进行操作。客户端向Web应用发出请求,对被测Web应用进行两种方式的检查:一是对Web服务器的响应进行检查;二是对数据库进行数据检查。任一个客户端中可定义多个检查项目。

引擎设计见图1,ScriptGenerate类提供了创建XML测试脚本的方法,ParseXML类实现了对测试脚本的解析操作,ExecuteThread类是测试执行的核心,根据脚本中的定义可实例化多个测试执行线程,WebAnalyse类给出了对Web应用中各类检查点的检查方法,例如链接检查、表单检查等[5]。

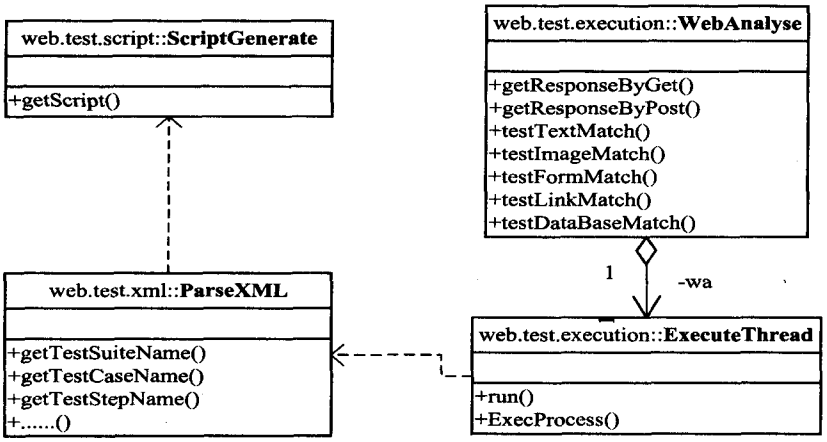

图1　测试执行引擎类图

3　测试脚本

测试脚本一般指可被自动化测试工具执行的针对一个特定测试的一系列指令,由测试执行引擎解析和执行,因此是自动化测试实现中的关键部分。

测试脚本可使用测试自动化工具自动生成(见图2),或手工创建(记录),或用编程语言编程来完成,也可综合前三种方法来完成。

本引擎采用基于XML的测试脚本[3],由脚本生成器自动生成(见图2),结构清晰易读,测试人员可轻松编辑及维护。测试脚本的生成是基于模型的软件测试方法:测试模型(Test Model)是对现实的一种抽象,它是建模者站在实际角度对现实事物的某些特征进行提取,精炼后得到的一个能代表原事物而又较原事物更容易分析、把握的对象。根据相应的测试准则和测试模型通过测试用例生成器自动生成测试用例,然后通过脚本生成器把测试用例转换成测试脚本。

脚本拥有自己的语法规则和保留字,并采用数据驱动方式将测试输入存储在独立的数据文件中而不是绑定在脚本中,从而提高了脚本复用率,降低了测试开销。脚本定义测试用例、测试步、测试输入和检查点及预期输出等信息,并通过检查点来判断结果是否正确。测试脚本主要定义测试用例及其输入和预期输出,由测试包(TestSuite)、测试用例(TestCase)、测试步(TestStep)三层结构嵌套而成[7]。

测试包:每个测试脚本包含一个测试包,一个测试包包含一个或多个测试用例,测试包为用来确认被测系统逻辑错误的测试用例的集合。

测试用例:一个测试用例包含一个或多个测试步,是完成一个明确的测试目标的测试,即具体条件下对Web应用某个场景的具体行为。

测试步:与Web应用的一次交互行为,即一次HTTP请求定义和一次预期HTTP响应定义[7]。

4　测试执行引擎的实现

测试执行引擎协调测试脚本、测试数据及被测Web应用三个对象,动态地调用脚本解析器和测试数据来实例化每个测试场景,通过判断响应页面中的检查点来对Web应用进行测试。脚本可根据测试人员对抽象序列的描述,通过脚本生成器自动获得;脚本解析器在执行过程中需多次被使用,因此以类的形式封装了所有所需方法作为一个辅助类;测试执行使用多线程技术并行地来对不同的测试场景进行测试,并记录测试日志[8]。

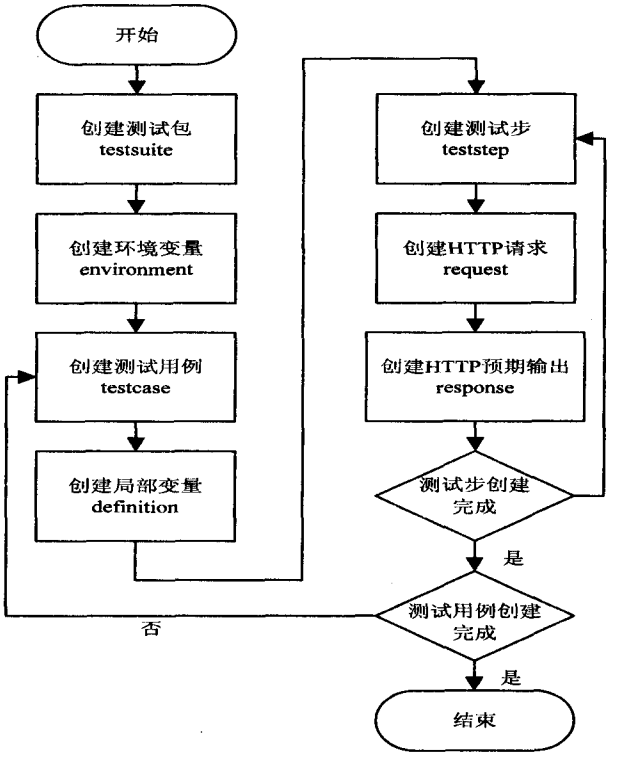

图2　脚本生成器流程图

4.1 脚本生成器

经过对抽象测试序列的描述，实例化后的测试用例信息记录于四个多维数组中，分别为测试序列(seqs)、状态(states)、迁移输入(event_inputs)和迁移预期输出(event_outputs)。然后程序根据记录的信息调用 ScriptGenerate 类来自动化生成脚本，生成后的测试脚本以 XML 格式保存用于测试复用。其实现流程见图 2。

4.2 脚本解析器

脚本解析类 ParseXML 作为测试执行的辅助类，提供了对测试脚本各个节点和属性值的提取，将在测试执行时被动态地调用[9]。解析器基于 Dom4J 开发，能方便地对 XML 格式文本进行分析。其类结构见图 3。

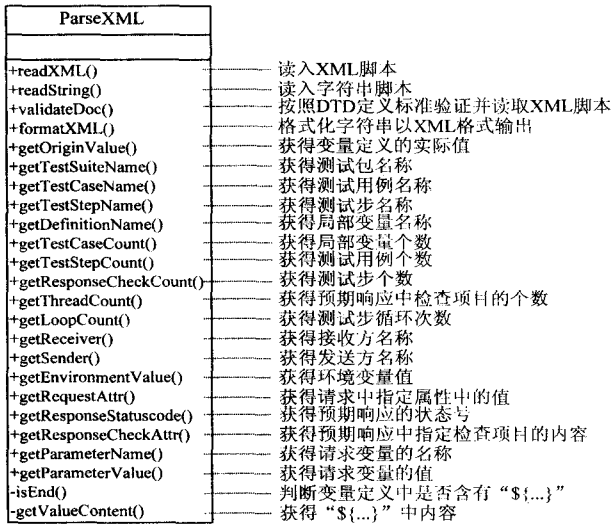

图 3 脚本解析类 ParseXML 类图

4.3 测试执行

在上述测试脚本的解释类的辅助下，测试执行引擎按一定顺序解析测试脚本，按照脚本中测试场景(即脚本中 testcase 节点)的个数实例化一个或多个线程，即模拟多用户来对具体 Web 应用进行并行测试。由于该测试执行引擎基于数据驱动方式，可以减少需要执行所有测试数据所需要的测试脚本量，即可以使用很少的脚本来产生大量的测试用例。例如，在一个测试场景中有 5 个顺序执行的测试步，其中两个测试步用到了外部数据文件作为测试输入，假设测试步 2 用到了 3 组测试数据；测试步 4 用到了 4 组数据，则最终该测试执行引擎会生成 12 个具体测试用例来覆盖所有的可能的测试数据组合。执行引擎会组合所有的可能，把不同的测试数据引入同一个场景来产生多种测试用例，通过 HTTPUnit 测试工具来对指定的 Web 应用进行测试。测试执行核心伪代码[10]如下：

```
private void run() {
    根据具体测试数据对该测试场景生成所有可能的测试用例
    Foreach 测试用例 {
        Foreach 测试步<teststep> {
            根据辅助类 ParseXML 中定义的方法来提取测试所需数据;
            提取测试步<teststep>标记中各属性值 name、loop 等;
            提取 HTTP 请求<request>标记中属性值 url、method 及所需提交参数 parameter;
            提取预期 HTTP 相应<response>标记中 statuscode 及所需检查的内容 check;
            使用 HTTPUnit 测试工具对 Web 应用发出请求并获得响应;
            调用辅助类 WebAnalyse 中对 Web 应用或数据库检查的方法来检查响应;
            输出 Web 测试日志;
            设置测试输出关键信息 reportDetails 用于测试报告; } } }
```

在上述测试执行过程中还用到了辅助类 WebAnalyse，其通过 HttpUnit 包提供的 API 实现了 6 种对 Web 应用功能测试的方法。分别通过 testTexMatch( )测试文本匹配；testImageMatch( ) 测试图像匹配；testFormMatch( ) 测试表单匹配；testLinkMatch( )测试链接匹配；testDatabaseMatch ( ) 测试数据库匹配；testTableMatch( )测试表格匹配[11]。

5 测试执行引擎的应用

上述几节描述测试执行引擎的原理、设计和实现方法，本节将通过一个 Web 应用实例来介绍引擎的实际应用。其前提是测试序列实例化及测试数据生成等工作的完成。

5.1 测试场景介绍

被测 Web 应用是一个基于 PHP 脚本及 MySQL 数据库的图书馆后台管理系统，其核心功能包括：读者管理、图书档案管理、图书借还管理等。限于篇幅，本节将仅对该系统的图书档案管理中图书添加功能进行测试。其对应 3 个网页 book. php(列举出已有图书)，book_add. php(添加图书界面)，book_ok. php(处理图书添加逻辑部分)，后台数据表为 tb_bookinfo。

上段文字描述被测系统中的图书添加功能，代表了一个测试流程，即一个测试场景。测试工具将模拟一个用户对被测系统的指定功能进行测试，测试场景为：管理员登录->首页->查看已有图书->进入图书添加页面->添加成功并返回已有图书。根据前提条件，在完成对测试序列中状态和迁移的具体化描述后，程序生成自动化测试脚本。测试执行工具分为三个区域：

1. 测试脚本编辑区域。测试人员可手工编辑 XML 测试脚本。

2. 在执行过程中，若缺少参数值会提示测试人员输入。

3. 测试执行日志信息输出。

为了体现出该自动化执行工具基于数据驱动的优

越性,测试场景在添加图书信息时使用了外部CSV数据文件(数据格式见表1),因此场景会被实例化成3个具体的测试用例进行测试,即同一个脚本根据不同输入数据而被执行了3次,因而达到了脚本复用的目的[12]。

5.2 测试报告

通过分析测试执行结果,测试工具自动生成基于XML格式的测试报告,报告使用XSLT(扩展样式表转换语言)进行样式格式化,并可由浏览器进行查看。报告以表格的形式分别描述了两个测试场景的执行情况,测试人员可查看预期输出网页特征检查的具体情况以及追踪测试步路径。实验对上表测试用例"AddBook"以3组数据进行测试,在测试步"Save1"中均出现了数据库写入错误,而测试用例"ModifyReader"中4个测试步均执行通过。

6 结束语

文中给出了测试执行引擎的工作流程,并描述了其重要组成部分,即测试脚本的设计。此外,还实现了自动化测试工具中脚本生成、解析和执行的方法,最后用一个具体测试场景描述本测试引擎的应用过程。

表1 测试场景测试输入数据

BARCODE	BOOKNAME	TYPEID	AUTHOR	TRANSLATOR	ISBN	PRICE	PAGE	BOKKCASEID
13746	实用软件测试	2	飞思科技	飞思科技	7-115	45	444	52
24157	自动化软件测试	2	陆璐	陆璐	7-115	40	359	52
48791	NetBeans	5	Peter	Peter	7-111	60	510	47

参考文献:

[1] 许　蕾,徐宝文. 应用测试框架研究[J]. 东南大学学报,2004,34(6):751-755.

[2] 周晓宇,许　蕾,徐宝文,等. Web应用的自动测试[J]. 计算机科学,2005,32(1):125-127.

[3] 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):94-95.

[4] Andrews A A, Offutt J, Alexander R T. Testing Web applications bymodeling with FSMs[J]. Softw Syst Model, 2005, 4: 326-345.

[5] Yang Ji-tzay, Huang Jiun-long, Wang Feng-jian, et al. Constructing an Object-Oriented Architecture for Web Application Testing[J]. Journal of Information Science and Engineering, 2002, 18: 59-84.

[6] 朱　菊,王志坚,杨　雪. 基于数据驱动的软件自动化测试框架[J]. 计算机技术与发展,2006,16(5):68-70.

[7] Jia Xiaoping, Liu Hongming, Qin Lizhang. Formal Structured Specification for Web Applications Testing[C]//2003 Midwest Software Engineering Conference. Chicago, USA: [s. n.], 2003.

[8] 赵剑英,王　镭. Web的自动化测试[D]. 天津:天津大学,2006.

[9] Liu Chien-Hung, Kung D C, Hsia P, et al. Structural Testing of Web Applications[C]//11th International Symposium on Software Reliability Engineering. [s. l.]: [s. n.], 2000.

[10] 路晓丽,郝克刚. Web应用软件的测试技术研究[D]. 西安:西北大学,2004.

[11] 许　蕾,徐宝文,陈振强. Web测试综述[J]. 计算机科学,2003,30(3):100-105.

[12] 贾晓霞,刘　和,吴　际,等. 一个基于下推自动机的Web测试自动执行器[J]. 计算机科学,2004,33(4):269-273.

(上接第27页)

[10] 许智宏,孙济洲. 基于蚂蚁算法的网格计算任务调度方法设计[J]. 天津大学学报,2004,37(5):414-418.

[11] 许智宏,孙济洲. 用蚂蚁算法进行网格任务调度的研究[J]. 计算机应用,2005,25(10):2236-2237.

[12] 赵晨阳,杨　裔,刘　礼,等. 基于蚂蚁算法的网格作业调度研究[J]. 微电子学与计算机,2007,24(9):78-81.

[13] 陈　歆,罗四维. 基于蚂蚁算法的网格任务分配算法研究[J]. 计算机技术与发展,2006,16(3):98-100.

[14] Yan Hui, Shen Xue-qin, Li Xing, et al. An improved ant algorithm for job scheduling in grid computing[C]// Proceedings of 2005 International Conference on Machine Learning and Cybernetics. [s. l.]: [s. n.], 2005: 2957-2961.

[15] Chang Ruay-Shiung, Chang Jih-Sheng, Lin Po-Sheng. An ant algorithm for balanced job scheduling in grids[J]. Future Generation Computer Systems, 2009, 25(1): 20-27.

[16] Buyya R, Murshed M. GridSim: a toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing[J]. Concurrency and Computation: Practice and Experience, 2002, 14: 1175-1220.