第17卷 第7期
2007年7月

计算机技术与发展
COMPUTER TECHNOLOGY AND DEVELOPMENT

Vol.17 No.7
Jul. 2007

自动化测试脚本自动生成技术的研究

蒋 云,赵佳宝

(南京大学 工程管理学院,江苏 南京 210008)

摘 要:软件自动化测试技术的出现,大大减轻了软件测试人员的测试压力,显著提高了测试工作的效率,但是自动化测试脚本的编写和维护工作也是测试自动化所面临的一项挑战。针对自动化测试脚本的编写和维护工作中遇到的困难,提出了一种基于RFT的测试脚本自动产生的方法,从而有效地提高了测试工作的效率,降低编写和维护测试脚本的开销。

关键词:软件测试;测试自动化;功能测试;回归测试

中图分类号:TP311.5 **文献标识码**:A **文章编号**:1673-629X(2007)07-0004-04

An Approach to Automatic Generation of Test Automation Scripts

JIANG Yun, ZHAO Jia-bao

(School of Management and Engineering, Nanjing University, Nanjing 210008, China)

Abstract: Software testing becomes easier and more efficient when test automation technique is applied to a test project. However, programming for test automation is also a burdensome task for testers. This article proposes an approach to automatically generate test automation scripts based on RFT, which can help testers to finish the test scripts quickly and reduce the cost of script maintenance.

Key words: software test; test automation; functional test; regression test

0 引 言

随着软件开发周期的时间限制、资源限制以及软件的日趋复杂化,对软件测试工作也提出了严峻的挑战。一个可靠稳定的产品仅靠手工测试已不能完全胜任,尤其是在迭代开发过程中,需要不断验证每一个版本下的各个组件都仍然保持原来的特性,因此必须要对其进行必要的回归测试(Regression Test)[1,2]。而传统的手工测试是一个劳动密集型的工作,如果每出一个新版本就让人工去做那些重复的、繁琐的、乏味的回归测试,必然是一项烦人的工作,而且由于测试人员的疏忽可能漏过一些回归错误。

因此,为了更好地节约人力物力,并让测试人员有更多的时间去发现新的问题,引入自动化测试已经成为各大软件公司的明智选择。然而,自动化测试在前期的准备工作中,需要进行脚本编制工作,而测试脚本的编写对于测试人员来说也是一项较为繁琐的工作,尤其是每个测试用例有大量重复的代码时,测试脚本的编写就变得枯燥乏味。笔者将就自动化测试脚本的编写进行一些研究,并基于测试工具Rational Functional Tester(IBM公司推出的一款功能测试工具)提出一个自动生成测试用例代码的方法,从而减轻测试人员的测试压力,并能提高测试代码的完整性和正确性。

1 自动化测试

测试自动化[3]是现在软件测试领域的一个重要组成部分,它可以代替大量的、重复的手工测试,可以降低人为的操作失误,减少测试工作量,提高测试效率,从而节约测试项目的成本。由于自动化测试的可重复性,因此它是回归测试的一个重要手段。那么如何来实现自动化测试呢,现行的主要方式就是通过编程实现,也就是说,通过自动测试工具(比如Mercury Interactive的Winrunner、QuickTest, IBM Rational Functional Test、Rational Robot等)或者程序语言(如Java等)编写出自动化测试脚本,通过脚本来控制被测软件(AUT, Application Under Test)各个对象,从而代替人为的操作流程,完成测试工作。当然自动化测试不是放之四海而皆准的,虽然它能够每次按部就班地进行同样的测试,但是它不具有测试人员的内在的知识和测试灵感,因此无法发现被测软件新的bug。另外,一些测试用例是无法用自动化脚本实现的,就必须用手

收稿日期:2006-11-17

作者简介:蒋 云(1982-),男,浙江诸暨人,硕士研究生,主要研究方向为管理信息系统、企业信息化、软件工程;赵佳宝,副教授,博士,主要研究方向为企业管控一体化、控制与系统工程。

工进行测试。综合上面几方面的考虑,在真正的测试项目中,通常是手工测试和自动测试想结合来完成一个测试任务,对于一些比较规范、功能稳定的模块,通常在经过一遍手工测试之后录制成自动化脚本,以后进行回归测试的时候,都通过自动化测试来完成,从而减少一部分测试工作量[4]。

2　RFT 以及分层测试理念

Rational Functional Tester(RFT)是一款先进的、自动化的功能和回归测试工具,它适用于测试人员和 GUI 开发人员[5]。RFT 具有基于向导(Wizards)的数据驱动的功能测试能力。在功能测试脚本的录制过程中,可以方便选择被测应用图形界面上的各种被测对象,进行参数化,通过生成新的数据池字段或从数据池中选择已存在数据字段,实现数据驱动的功能回归测试。然而这种测试对象与测试用例的紧耦合性会给测试工作带来一定的麻烦。由于软件开发过程中,GUI 元素会发生一些变化,而操作流程的变动却不大,此时,由于业务流程脚本和原有的测试对象图(用于记录 AUT 的被测对象)在包含在同一资源当中,原有的 GUI 对象已经不能够使用,所以需要更新对象和业务脚本,当测试对象改动比较大、数量比较多的时候,测试脚本的维护开销会成倍增加。为了解决这样的问题,需要提出一种分层测试框架[6],把测试对象脚本和业务脚本分离开,在测试对象发生改动时,只需要重新维护测试对象图,对业务脚本不产生变更或产生少量的变更,从而大大节约脚本的维护开销。

一般的分层框架主要分为两层:测试对象层(testObjects)和测试用例层(testCases)。测试对象层主要负责测试对象的抓取、更新以及为上一层提高调用对象的接口;测试用例层用来管理根据测试计划编制的测试用例,在业务上实现测试流程,通过测试对象层提供的对象调用接口来操控测试对象,从而实现与测试对象的松耦合。在实际的测试应用项目中,在测试对象层和测试用例层中间还会加上一层,用来实现和管理一些在测试用例中经常用到的方法(如图 1 所示),并向上给用例层提供接口。

文中提出的用例自动生成方法就是基于这种分层测试框架之上的,通过实际的检验,给测试工作带来了巨大的帮助。

3　测试用例的自动生成方法

此文是在实际应用测试项目实践的基础之上形成的,文中论述的 AUT 是基于 Eclipse 开发的一个 C/S 结构的应用程序,因此采用同样基于 Eclipse 平台的 RFT 作为测试工具,能够实现开发与测试基于统一平台,从而更好地达到测试效果。

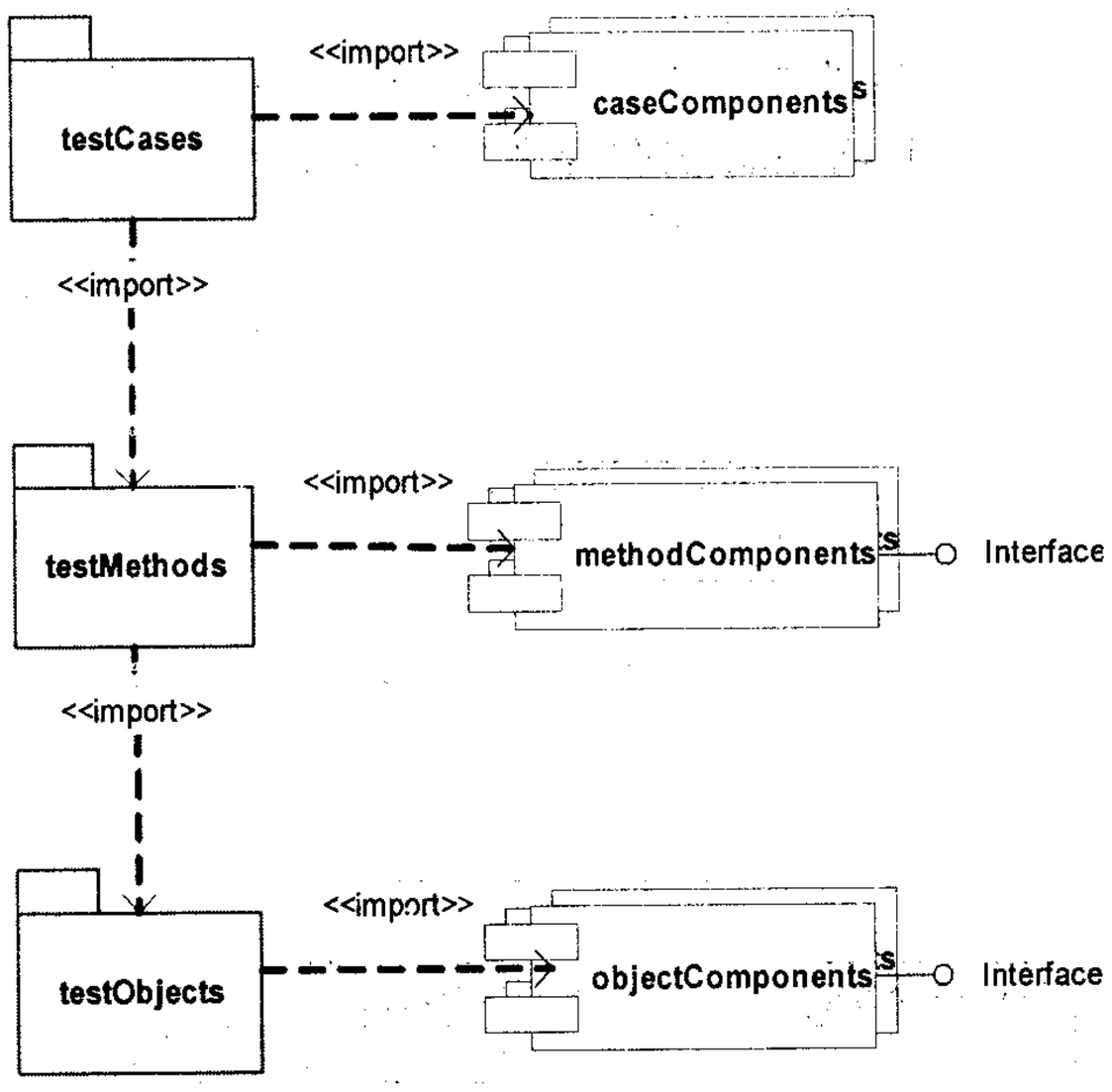

图 1　三层测试框架结构示意图

在实际的项目中,AUT 将一些包含着 Rough Data 的文本文件进行数据解析,形成了一个 XML 文件,并把结果用图形化显示在 GUI 上,提供给用户更加清晰的数据结构。因此,有一部分测试工作需要对显示在 GUI 上的数据结构进行核对,从而来验证其中的转换算法是否正确。因此在实际的测试过程中,需要先对 GUI 上的数据结构与原始的、包含 Rough Data 的文本文件中的数据结构以及值进行手工比较,从而找到一些测试数据,并把这些通过人工测试找到的正确的测试数据保留下来,用于自动化测试的脚本中,然后再把这些自动化测试脚本用于回归测试。通过这种手工测试和自动化测试相结合的方式,既可以保证测试数据的准确性,也可以利用自动化测试脚本进行回归测试,减少不必要的手工劳动,在实际的测试实践中,这种方法被证明是有效的。

虽然这种方式被证明是有效的,但是在编写自动化测试脚本的时候,由于每个测试用例包含的测试数据相当庞大,一个测试用例需要平均比较大约 400～500 个数据值,因此测试脚本的编写工作将会变得相当的繁重。而在采用三层测试框架的基础上,测试数据的比较工作只需要一个在 testMethods 包里定义,如 compareValue(String valueName, String actualValue)的方法来完成,因此在 testCases 层的测试脚本不需要了解如何从 GUI 获取界面上显示的数据的细节,而只需要提供正确的数据值和被比较数据的名称即可,然后调用 compareValue 方法即可完成测试脚本,由于测试数据是通过手工测试验证得到的静态数据,因此,基于

这样的环境,笔者提出了一种自动生成测试脚本的方法,来代替手工编写测试脚本,从而提高测试的效率,节省测试人员的时间。

在实际的项目中,一个 XML 文件会包含多个 Diagram,每个 Diagram 又包含多个 Node,而一个 Node 包含一个 Descriptor,而每个 Descriptor 中就包含着需要测试的数据。数据结构如图 2 所示,其中 attribute 属性所包含的内容就是需要测试的数据的信息。

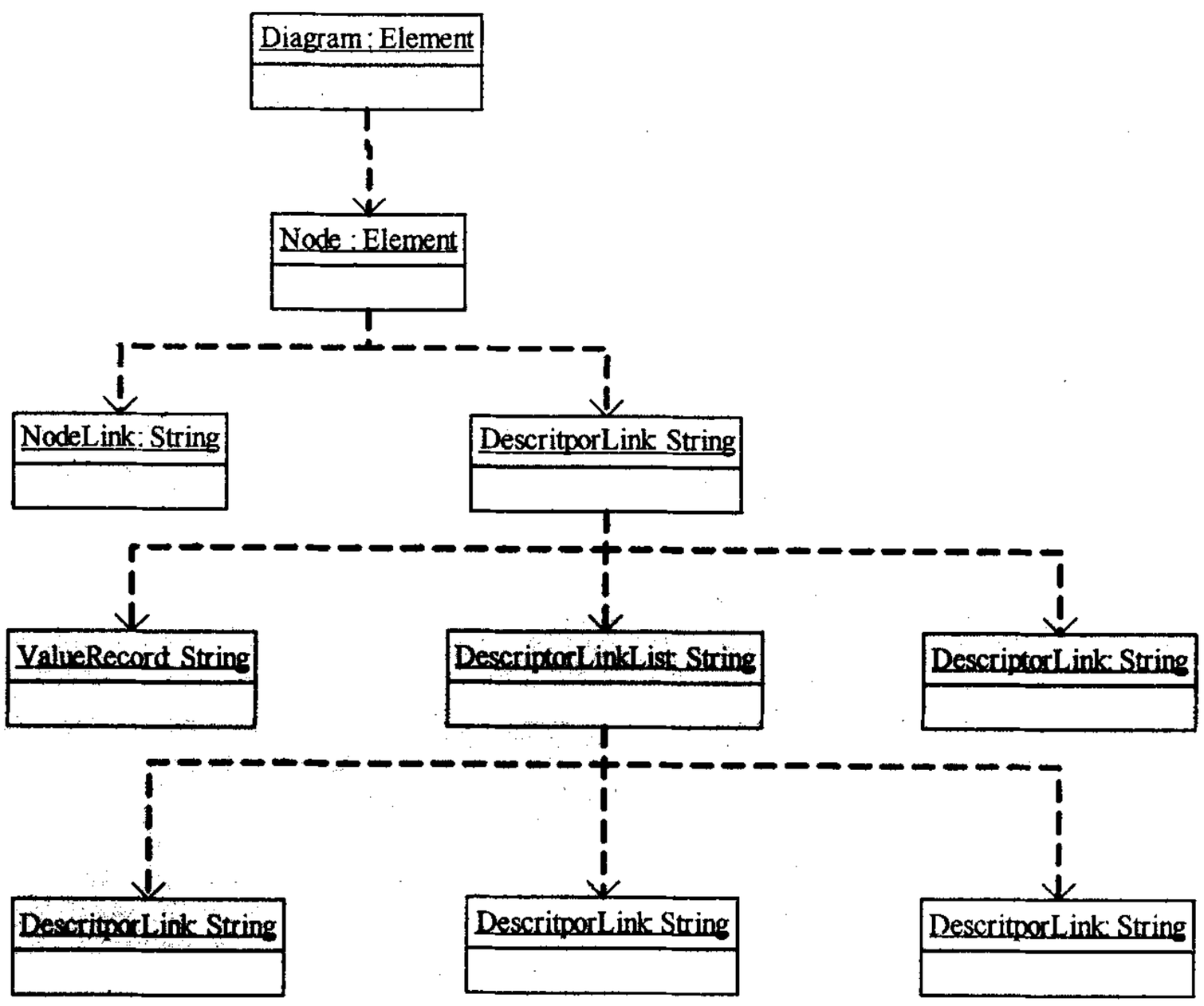

图 2 被测试数据层次结构

基于这样的结构,设计了两个类:ParseXML 和 CaseGenerator 来完成脚本自动生成的功能。其中 ParseXML 用来将需要测试的数据名称和实际值从原始的 XML 文件中解析出来;CaseGenerator 用来根据测试脚本的要求生成相应的脚本。类图如图 3 所示。

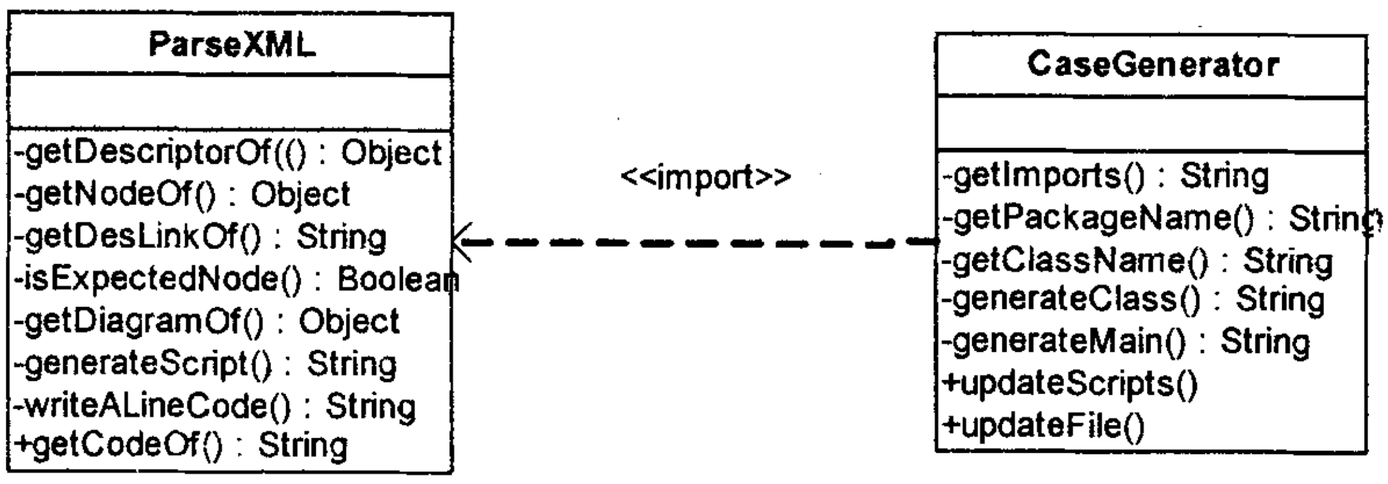

图 3 ParseXML 与 CaseGenerator 类结构与关系

通过被测数据结构图,可以看到每个 Descriptor 不仅包含要测试的数据,而且还可以指向另外一个 Descriptor,而这个新的被指定的 Descriptor 中的数据也是需要连同测试的。因此在程序 generateScript 方法中需要通过递归来实现。

通过解析 XML 文件获取数据,进而产生代码的主要流程如图 4 所示。

ParseTree 类中获取数据值并产生代码的流程为:

(1)通过外部参数传入的 Diagram 名称在 XML 文件中定位到相应的 Diagram 对象。

(2)通过外部参数传入的 Node 名称列表在 Diagram 中找到对应的 Node 对象。

(3)通过外部参数传入的 DescriptorID 列表调用 getDescriptor 方法获取 Descriptor 对象。

(4)获取 Descriptor 对象的所有 Attribute 子节点 List。

(5)如果 Attribute[i]节点包含所要验证的数据值信息,则通过 writeALineCode 方法产生一条比较代码。如果 Attribute[i]包含子节点,而且该节点是 DescriportLinkList 类型的,则转入(6);如果 Attribute[i]包含子节点,而且该节点是 DescriportLink 类型的,则获取 DescriptorID 并转向(3);如果 i 等于 List. length,转入(7)。

(6)获取 DescriptorLinkList 所拥有的所有 DescriptorLinkIDs,依次把 DescriptorLinkIDs[j]作为传入参数,转入(3),直到 j 等于 DescriptorLinkIDs. length。

(7)流程结束。

最后通过一个 getCodeOf 方法把用户需要验证的 Node 的所有比较代码整理返回,以提供给 CaseGenerator 中的方法调用,并最终更新 RFT 代码。

在另外一个类 CaseGeneratotr 中,updateScript 方法是暴露给外部客户使用的,CaseGenerator 类中各个方法的调用关系如图 5 所示。用户在 RFT 环境下文中所提出的方法自动生成脚本的时候,可以依照以下步骤进行:首先,需要在 RFT 中添加一个空脚本(如 Case1),即测试用例的自动化脚本。然后再新创建一个用于更新此脚本的类,这个类可以仿造添加空脚本的方法直接在 RFT 中产生,为方便起见,称之为 CaseUpdator。在 CaseUpdator 中,用户需要传入几个主要的对象参数:Case1 的一个实例,用户所要验证 NodeList,可以使用 Vector 的实例也可以是 Hashtable 的实例,只要与 updateScript 方法中的参数列表相对应即可。方法参数列表可以根据实际需求进行定义。

文中所论述的测试项目中,CaseUpdator 中 test-

Main 中调用 CaseGenerator. updateScript 方法,然后运行脚本便可以更新 Case1 的脚本。

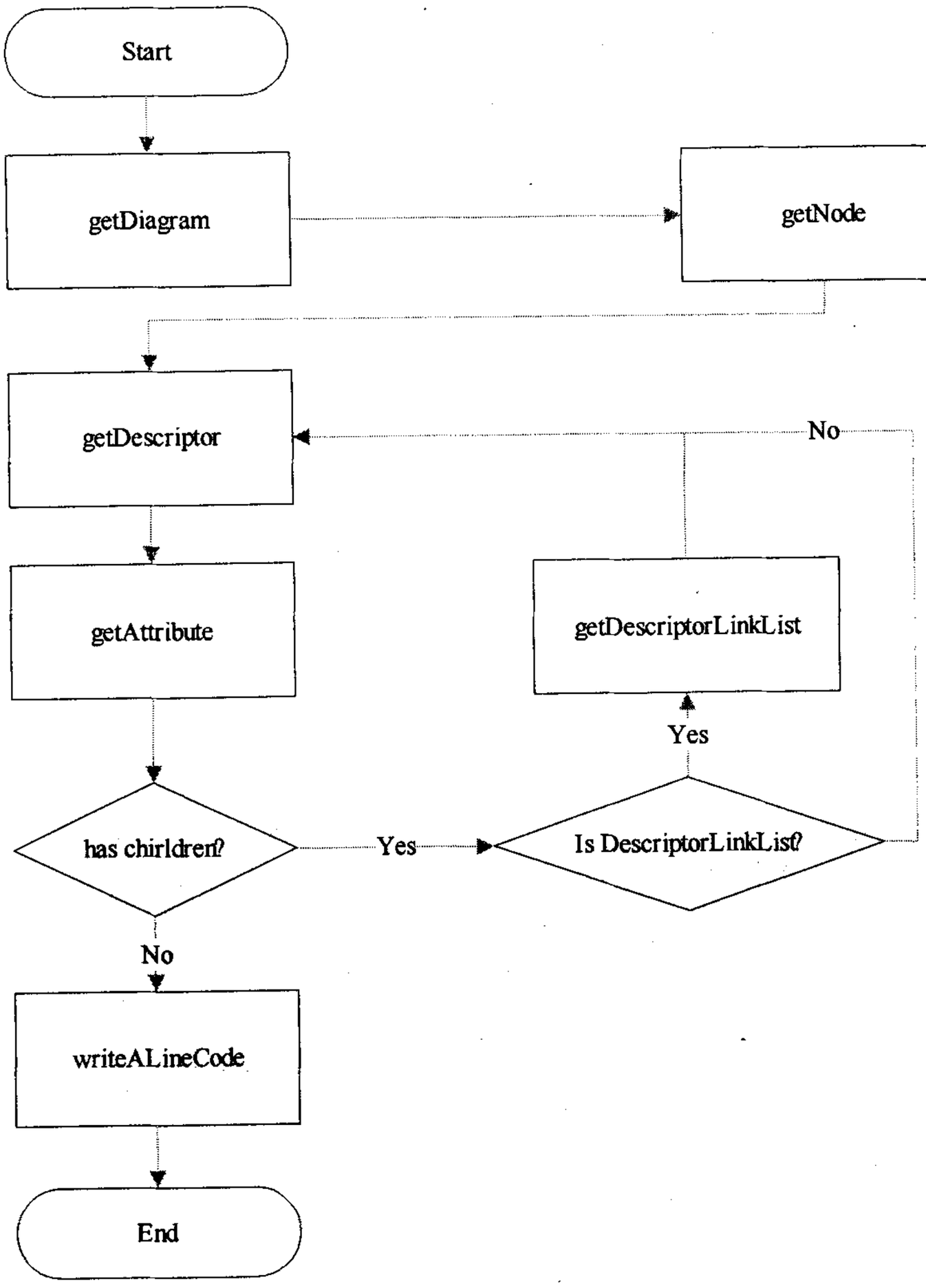

图 4　解析文件生成代码的流程

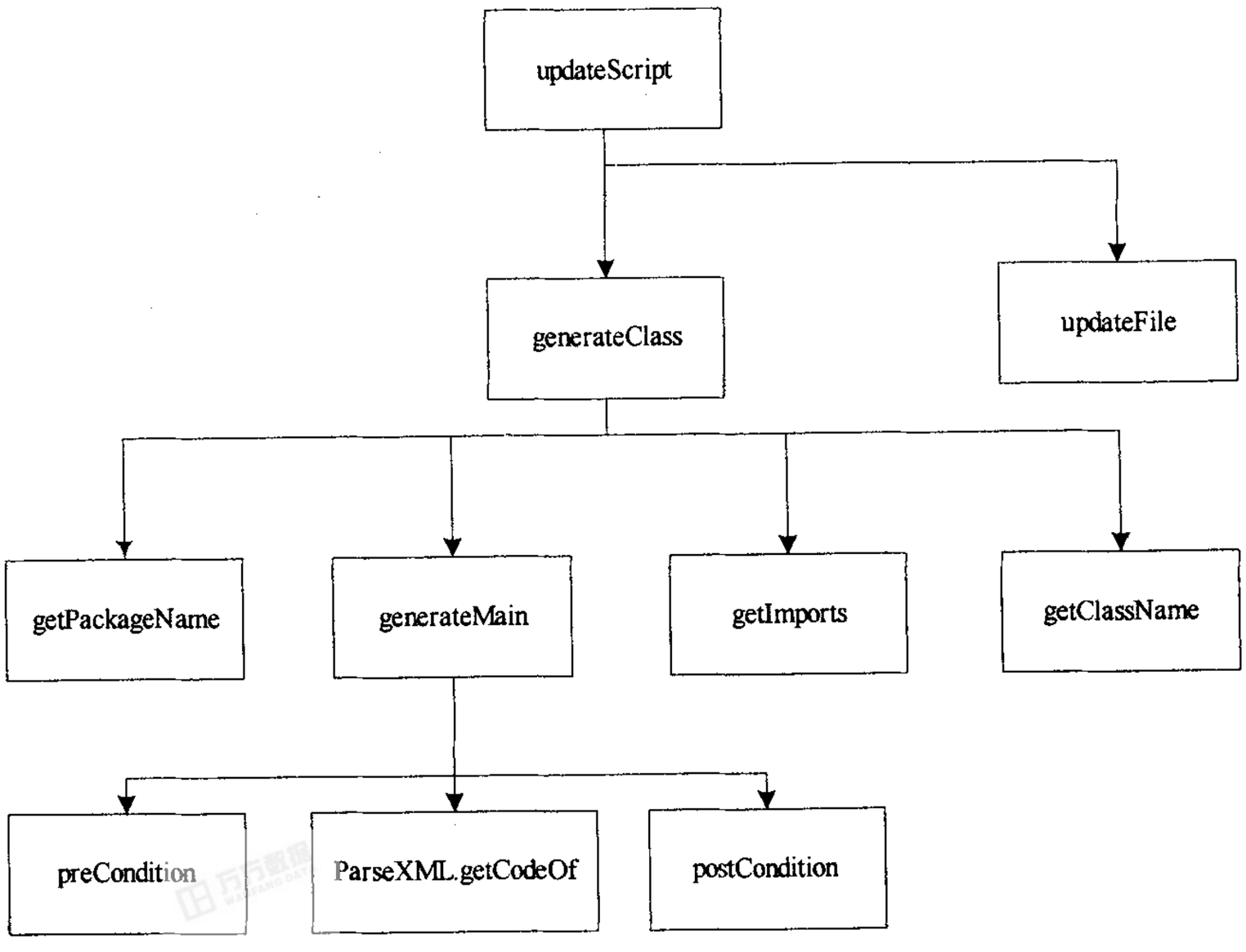

图 5　CaseGenerator 类中方法调用关系

4　结束语

自动化测试方法的应用大大提高了软件测试工作的效率,尤其是在回归测试中,把一些重复的、机械的测试任务交给计算机去做,很大程度上减轻了测试人员的压力,并在一定程度上提高了回归测试的准确性。当然,自动化测试也存在着一些缺陷:自动化测试不能发现软件新出现的问题,不能根据实际的情况及时调整测试流程,而且自动化测试脚本的编写和维护工作也是一项比较繁杂的工作。文中就自动化脚本编写中遇到的问题,提出了一种基于 RFT 的脚本自动生成方法,在一定程度上减轻了测试人员编写自动化测试脚本的压力,提高了测试工作的效率,增强了脚本的可维护性,当用户需要测试的 Node 进行变动时,利用生成工具可以快速地更新测试脚本。当然,文中所论述的方法具有一定的局限性,比如说平台的局限性以及测试需求的局限性。通过文中所论述的思想,利用分层测试的理念把自动化脚本的自动生成方法利用到更多的测试平台、更多的测试项目当中,从而能够减轻测试人员的负担,提高测试工作的效率。

参考文献:

[1] 尤永康,刘乃琦. 自动化回归测试在 java 项目中的实现[J]. 计算机应用,2005,25(1):88－90.

[2] 马雪英,姚　砺,叶澄清. 回归测试自动化工具研究[J]. 计算机科学,2005,32(3):162－165.

[3] 陈计喜,姜丽红. 自动化功能测试的方法与实现[J]. 计算机工程, 2004, 30(12):168－169.

[4] 宋　波,张忠能. 基于系统功能测试的软件自动化测试可行性分析[J]. 计算机应用与软件,2005,22(12):31－33.

[5] 宁德军. Rational 完成自动化功能测试[EB/OL]. 2005. www. 51testing. com.

[6] Pettichord B. Deconstructing GUI Test Automation[J/OL]. STQE, 2003, (1/2)[2003]. www. stqemagazine. com.