[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06F 11/36 (2006.01)
G06F 9/44 (2006.01)

[12] 发明专利说明书

专利号 ZL 200710176970.7

[45] 授权公告日 2009 年 10 月 14 日

[11] 授权公告号 CN 100549978C

[22] 申请日 2007.11.7
[21] 申请号 200710176970.7
[73] 专利权人 中兴通讯股份有限公司
地址 518057 广东省深圳市南山区高新技术产业园科技南路中兴通讯大厦法务部
[72] 发明人 梁 君
[56] 参考文献
US20050223362A1 2005.10.6
CN1710547A 2005.12.21
CN1873626A 2006.12.6
审查员 韩倩倩

[74] 专利代理机构 北京银龙知识产权代理有限公司
代理人 许 静

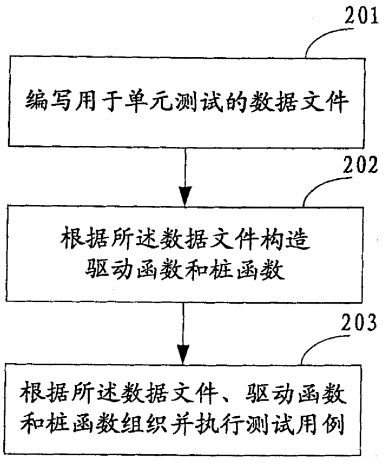
权利要求书 1 页 说明书 8 页 附图 3 页

[54] 发明名称

一种数据驱动的单元测试方法

[57] 摘要

本发明公开了一种数据驱动的单元测试方法，包括如下步骤：A、编写用于单元测试的数据文件，所述数据文件包括配置数据文件和报文数据文件；B、根据所述数据文件构造驱动函数和桩函数；C、根据所述数据文件、驱动函数和桩函数组织并执行测试用例。 本发明的方法采用数据文件存储测试数据，使测试数据重用率高，便于管理；通过测试数据驱动测试用例的执行，对测试执行的控制灵活，开发和测试工作可分离进行。



1. 一种数据驱动的单元测试方法，其特征在于，包括如下步骤：

A、编写用于单元测试的数据文件，所述数据文件包括配置数据文件和报文数据文件，其中，所述配置数据文件由测试数据和控制信息组成；

B、根据所述数据文件构造驱动函数和桩函数，其中，

构造驱动函数进一步包括：从配置数据文件中读取测试数据，将该测试数据作为被测函数的输入数据或者作为报文数据文件名；从报文数据文件中读取报文数据，将该报文数据作为被测函数的输入数据或者校验数据；将被测函数执行后的输出报文存储到报文数据文件中，用于验证执行结果，

构造桩函数进一步包括：从报文数据文件中读取报文数据，将该报文数据作为桩函数的输出数据或者校验数据；将被测函数执行后输入到桩函数中的报文存储到报文数据文件中，用于验证执行结果；

C、根据所述数据文件、驱动函数和桩函数组织并执行测试用例。

2. 如权利要求 1 所述的方法，其特征在于：

步骤 A 中，所述配置数据文件采用 XML 格式编写，并包括驱动函数名和为所述驱动函数配置的测试数据。

3. 如权利要求 2 所述的方法，其特征在于：

为所述驱动函数配置多组测试数据。

4. 如权利要求 1 所述的方法，其特征在于：

步骤 A 中，所述报文数据文件采用 VAL 格式，并利用报文数据文件生成工具生成或者按照报文格式手工编写。

5. 如权利要求 1 所述的方法，其特征在于，步骤 C 进一步包括：

上层驱动函数从配置数据文件中读取子驱动函数名，以此来决定需要执行的子驱动函数以及子驱动函数的执行顺序；

上层驱动函数从配置数据文件中读取测试数据组数，以此来决定子驱动函数的执行次数。

一种数据驱动的单元测试方法

技术领域

本发明涉及单元测试领域，特别是涉及一种数据驱动的单元测试方法。

背景技术

软件单元，指软件设计说明中一个可独立测试的元素，是程序中一个逻辑上独立的部分。单元测试是对单个的软件单元或一组相关的软件单元所进行的测试，检查各软件单元是否正确地实现了规定的功能。单元测试对于在开发早期充分发现程序漏洞，以及提高程序的稳定和健壮具有极其重要的意义。

单元测试的关键是构建测试环境，其中经常要用到驱动和桩。驱动是用于模拟待测单元调用者的程序，桩是用于模拟待测单元所调用的下层单元的程序。

在驱动和桩中经常要使用各种测试数据，有的是作为待测函数的输入，有的是作为桩函数的返回值，有的是作为待测函数输出数据的校验数据等。特别是在网管软件中，各模块之间存在着大量的报文数据交换，模块内处理的数据和输出的数据也大多是报文数据，因此，网管软件的单元测试需要构造各种报文数据。另外，在测试时，不可避免还需要其他类型的测试数据，如整型、布尔型或字符串数据等。

目前软件开发中的单元测试，大体分为三种：第一，用户手工编写桩和驱动函数的纯手工测试；第二，借助白盒测试开源代码工具，如 CppUnit，采用“半自动”的方式；第三，采用脚本写桩和驱动的自动测试。

对于自动化测试方法，通常采用读取数据文件来得到测试数据。该方法虽然灵活性提高，但对于网管软件中大量结构复杂的报文数据，如果全手工在数据文件中编写，不仅工作量大，而且难以表现报文复杂的层次结构，可移植性和重用率也不是很高。另外，自动化测试工具不仅需要用户熟悉脚本或某种语言进行测试逻辑控制，提高了对测试人员技能的要求，而且，通用的单元测试工具在特定的环境下，自动化程度不高，这限制了其在一些具体环境中的广泛

使用。

前两种测试方式使用简单，操作灵活，覆盖率易控制，在网管软件开发中应用广泛。但这两种方式的测试数据通常采用硬编码的方式直接在桩或驱动函数中编写，存在以下不足：

（1）数据构造量大而繁琐，大量的测试数据难以重用和管理；

（2）测试代码重用率低，每增加一组测试数据，都需要新增一组测试代码，容易造成代码冗余；

（3）灵活性差，每次增加或修改一组测试数据，都需要进行重新编译。

发明内容

本发明所要解决的技术问题是提供一种由外部数据文件提供测试数据并控制测试执行的单元测试方法。

为解决上述技术问题，本发明提供技术方案如下：

一种数据驱动的单元测试方法，包括如下步骤：

A、编写用于单元测试的数据文件，所述数据文件包括配置数据文件和报文数据文件；

B、根据所述数据文件构造驱动函数和桩函数；

C、根据所述数据文件、驱动函数和桩函数组织并执行测试用例。

较佳地，步骤 A 中，所述配置数据文件采用 XML 格式编写，并包括驱动函数名和为所述驱动函数配置的测试数据。

较佳地，为所述驱动函数配置多组测试数据。

较佳地，步骤 A 中，所述报文数据文件采用 VAL 格式，并利用报文数据文件生成工具生成或者按照报文格式手工编写。

步骤 B 中，根据所述数据文件构造驱动函数进一步包括：从配置数据文件中读取测试数据，将该测试数据作为被测函数的输入数据或者作为报文数据文件名；从报文数据文件中读取报文数据，将该报文数据作为被测函数的输入数据或者校验数据；将被测函数执行后的输出报文存储到报文数据文件中，用于验证执行结果。

步骤 B 中，根据所述数据文件构造桩函数进一步包括：从报文数据文件中读取报文数据，将该报文数据作为桩函数的输出数据或者校验数据；将被测

函数执行后输入到桩函数中的报文存储到报文数据文件中，用于验证执行结果。

步骤C进一步包括：上层驱动函数从配置数据文件中读取子驱动函数名，以此来决定需要执行的子驱动函数以及子驱动函数的执行顺序；上层驱动函数从配置数据文件中读取测试数据组数，以此来决定子驱动函数的执行次数。

本发明提供了完整的由外部数据文件提供测试数据并控制测试执行的单元测试方法，与现有技术相比，具有如下有益效果：

（1）采用数据文件存储测试数据，使测试数据重用率高，且便于管理；

（2）通过测试数据驱动测试用例的执行，利用测试数据组数控制用例执行次数，对测试执行的控制灵活，并提高测试代码重用率；

（3）对测试数据的修改灵活，无需重新编译代码；

（4）无需数据库的支持，减少对测试环境的依赖；

（5）对被测程序代码几乎没任何影响和依赖，开发和测试工作可分离进行。

附图说明

图1 为本发明数据驱动的单元测试系统结构示意图；

图2 为本发明数据驱动的单元测试方法流程图；

图3 为本发明的方法中组织并执行测试用例的流程图；

图4 为本发明的方法中解析XML文件的流程图。

具体实施方式

为使本发明的目的、技术方案和优点更加清楚，下面将结合附图及具体实施例对本发明进行详细描述。

请参照图1，本发明在进行单元测试时，所用的测试系统主要包括，数据文件、数据文件解析模块、报文数据读写接口和测试框架模块，其中：

数据文件包括配置数据文件和报文数据文件，配置数据文件由测试数据和控制信息组成，报文数据文件采用已被广泛接受的格式存储报文数据。数据文件解析模块用于分析配置数据文件，提取测试数据和控制信息，并装载好测试用例供测试框架模块执行。报文数据读写接口由报文数据读取函数和报文数据输出函数组成，报文数据读取函数用于从报文数据文件中读取报文信息，报文

数据输出函数用于将报文信息输出到报文数据文件中。测试框架模块用于驱动数据文件解析模块运行，并执行数据文件解析模块装载好的测试用例，以及初始化测试环境和清除测试环境。

请参照图 2，本发明数据驱动的单元测试方法包括如下步骤：

步骤 201、编写用于单元测试的数据文件；

（1）编写配置数据文件：

配置数据文件除了能提供测试数据，还能提供测试控制信息。配置数据文件按照预定的规范编写，以不同的关键字代表不同的配置内容，且各关键字都有开始和结束符来标识配置内容的范围，各关键字之间有从属关系。如用例名关键字下设置将要执行的测试函数，测试数据关键字下为当前测试函数配置一组测试数据。一个用例名关键字下可以有若干个测试数据关键字。

本发明的配置数据文件可以采用 XML 文件，其格式如下：

```
<?xml version="1.0">
<TestCaseConfig>
    <TestCase Path = "TestMethod">
        <TestData   Name1 = “Value1”   Name2 = “Value2”>
        </TestData>
        <TestData   Name1 = “Value3”   Name2 = “Value4”>
        </TestData>
    </TestCase>

    <TestCase Path = "TestMethod2">
        <TestData   NameA = “ValueA”   NameB = “ValueB”>
        </TestData>
        <TestData   NameA = “ValueC”   NameB = “ValueD”>
        </TestData>
    </TestCase>
</TestCaseConfig>
```

其中：

<TestCase>用来标识一个将要执行的测试用例（测试函数），其中的 TestMethod 就是一个测试函数名。

<TestData>用来标识一组测试数据，供当前<TestCase>标识的测试函数使用。每一个测试数据由数据名和数据值组成，其中的“Name1”代表数据名，

“Value1”代表数据值。<TestData>是<TestCase>下的元素，一个<TestCase>下可以有多个<TestData>，即每一个测试函数可以用多组测试数据。

<TestCaseConfig>没有实际含义，通过将它作为最顶层的元素，可使一个XML文件中包含多个<TestCase>。

（2）编写报文数据文件：

报文数据文件采用被业界广泛接受的VAL格式，可采用多种方式生成，如利用报文数据文件生成工具，或利用报文数据输出函数，或按照报文格式手工编写。

步骤202、根据所述数据文件构造驱动函数和桩函数；

（1）编写驱动函数，模拟被测函数的调用者，具体为：

（a）利用数据文件解析模块提供的测试数据读取接口函数来引用步骤201中准备的配置数据文件中的测试数据。接口函数的输入参数为数据名，返回值为数据值。通过不同的接口函数，可分别得到字符串型、布尔型、整型等不同类型的数据值。从配置数据文件读取的数据值有的作为控制测试环境的标志位，有的作为测试函数的输入数据，有的作为报文数据文件名等。

（b）利用报文数据读取函数从报文数据文件中读取报文数据，用于构造被测函数的输入数据或者校验数据。在传输网管软件中，通常使用ASN.1（Abstract Syntax Notation One）报文。通过扩展SNACC（一种开源的ASN.1文件编译器）编译器，使其在编译ASN.1文件时，生成报文数据读取函数ReadValFile和报文数据输出函数PrintVal，它们分别能读取和打印VAL格式（被业界广泛采纳的格式）的报文数据文件。根据在（a）中得到的报文数据文件名，调用ReadValFile函数来得到一个报文对象，用于构造被测函数的输入参数。

（c）利用报文数据输出函数将被测函数执行后的输出报文存储到报文数据文件中，用于验证执行结果，具体为：（c1）调用数据文件解析模块提供的数据读取接口函数来引用配置数据文件中配置的检验数据或报文数据文件名，利用检验数据构造验证代码；（c2）根据在（c1）中得到的报文数据文件名，调用ReadValFile函数来得到一个报文对象，作为被测函数输出值的检验数据，利用检验数据构造验证代码；（c3）根据在步骤（c1）中得到的报文数据文件

名，调用 PrintVal 来打印被测函数的输出报文，用作外部检验。

（d）利用测试框架模块对驱动函数进行注册。例如测试框架模块采用 CppUnit，利用 CppUnit 提供的 CPPUNIT_TEST 宏来注册驱动函数到测试用例组。

（2）编写桩函数，模拟被测函数所调用的下层单元，具体如下：

（a）利用数据文件解析模块提供的测试数据读取接口函数来引用步骤 201 中准备的配置数据文件中的测试数据；

（b）利用报文数据读取函数从报文数据文件中读取报文数据，用于构造桩函数的输出数据或验证数据；

（c）利用报文数据输出函数将被测函数传入桩函数中的报文数据存储到报文数据文件中，用于验证执行结果。

步骤 203、根据所述数据文件、驱动函数和桩函数组织并执行测试用例。具体为：

（a）数据文件解析模块装载配置数据文件；

（b）数据文件解析模块解析配置数据文件，根据配置数据文件中不同的关键字做出不同的处理：

（b1）如为用例名关键字的开始符，则解析得到测试用例名，并利用测试框架模块装载该测试用例；

（b2）如为测试数据关键字的开始符，则首先清除内存中原有的测试数据，然后读取该关键字下配置的所有测试数据，并将测试数据存放于内存中供测试数据读取接口函数使用；

（b3）如为测试数据关键字的结束符，则运行当前的测试用例；

（b4）如为用例名关键字的结束符，则卸载当前的测试用例。

请参照图 3，给出一个根据配置数据文件组织并执行测试用例的应用实例，其包括如下步骤：

步骤 301、启动数据文件解析模块；

在 main()函数中，调用数据文件解析模块的接口函数，开始运行数据文件解析模块。

步骤 302、创建并初始化数据文件解析模块中需要用到的对象；

其中的几个关键对象创建如下:

（1）创建并初始化 CTestCaseInvoker 类对象，CTestCaseInvoker 为测试用例调用者类，用于处理测试用例开始事件、测试用例结束事件和运行测试用例等;

（2）创建并初始化 CXmlTestCaseParser 类对象，CXmlTestCaseParser 类用于根据一个 XML 文件分析测试用例;

（3）创建并初始化 CXmlTestDataHandler 类对象，CXmlTestDataHandler 类用于解析和处理以 XML 文件保存的测试数据；数据文件解析模块对 XML 文件的分析是以 ACE（The Adaptive Communication Environment）为基础的，CXmlTestDataHandler 即继承至 ACE 的 ACEXML_DefaultHandler 类;

（4）创建并初始化 CPropertyHandler 类对象，CPropertyHandler 用于监听测试数据，并将测试数据保存在内存中。

步骤 303、装载 XML 文件。将 XML 文件名传给 CXmlTestCaseParser 类对象。

步骤 304、解析 XML 文件并执行测试用例;

CXmlTestCaseParser 对象调用 Parser 方法来解析 XML 文件，解析过程请参照图 4，具体如下:

（1）创建一个 ACE 中的 ACEXML_FileCharStream 类对象，用于打开 XML 文件。ACEXML_FileCharStream 类用于从一个文件中读取输入信息。

（2）将 ACEXML_FileCharStream 对象封装为 ACEXML_InputSource 类对象 input。ACEXML_InputSource 也是 ACE 中的一个类，它封装了实际的输入流及其他附加信息。

（3）创建一个 ACEXML_Parser 类对象 parser，并用 CXmlTestDataHandler 类对象对 parser 进行初始化。ACEXML_Parser 也是 ACE 的一个类，它实现了一个基于 SAX（Simple API for XML）的解析器。

（4）调用 parser.parse(input)方法。parse 为 ACEXML_Parser 类的一个方法，用于解析 XML 文件。input 为（2）中创建的 ACEXML_InputSource 类对象。ACEXML_Parser 的 parse 方法在解析 XML 文件中的元素时，会根据收到的不同通知引发不同的操作。

（5）如果接收到元素开始通知，执行 CXmlTestDataHandler:: startElement 方法，具体为:（51）如果当前处理的元素为“TestCase”，则解析 XML 文件得到测试用例名，并利用 CppUnit 框架加载该用例;（52）如果当前处理的元素为“TestData”，首先，清空内存中原有的测试数据，然后，从 XML 文件中读取当前 TestData 元素下的所有测试数据，并将数据保存在内存中，供数据读取接口函数来引用;（53）如果当前处理的元素为其他，则不作任何处理。

（6）如果接收到元素结束通知，执行 CXmlTestDataHandler:: endElement 方法，具体为:（61）如果当前处理的元素为“TestData”，则执行当前测试用例;（62）如果当前处理的元素为“TestCase”，则卸载当前测试用例。

步骤 305、当 XML 文件全部解析完，析构步骤 302 中创建的对象，清除框架环境，测试完成。

最后应当说明的是，以上实施例仅用以说明本发明的技术方案而非限制，本领域的普通技术人员应当理解，可以对本发明的技术方案进行修改或者等同替换，而不脱离本发明技术方案的精神范围，其均应涵盖在本发明的权利要求范围当中。

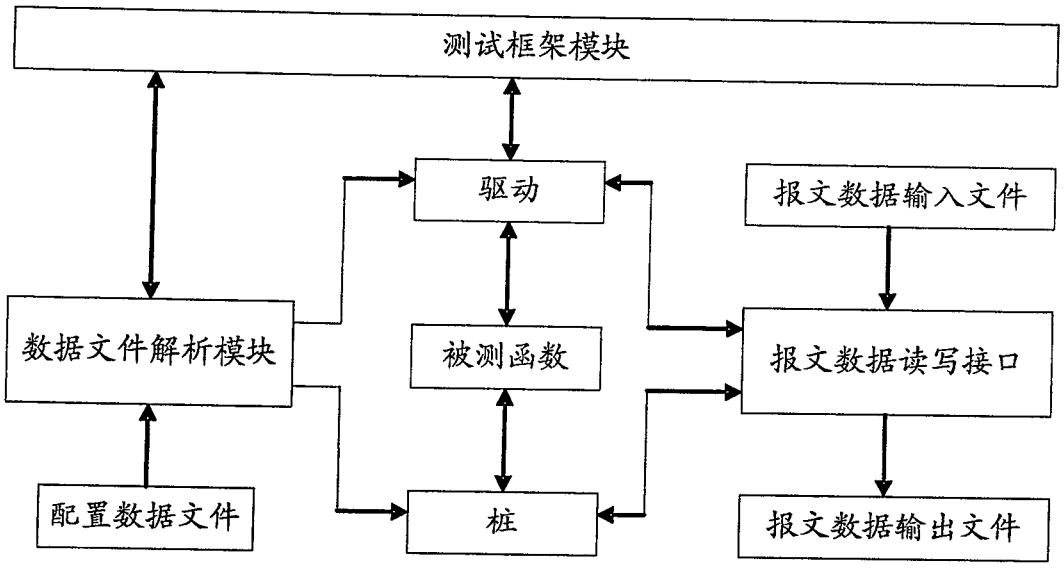

测试框架模块
驱动
报文数据输入文件
数据文件解析模块
被测函数
报文数据读写接口
配置数据文件
桩
报文数据输出文件

图 1

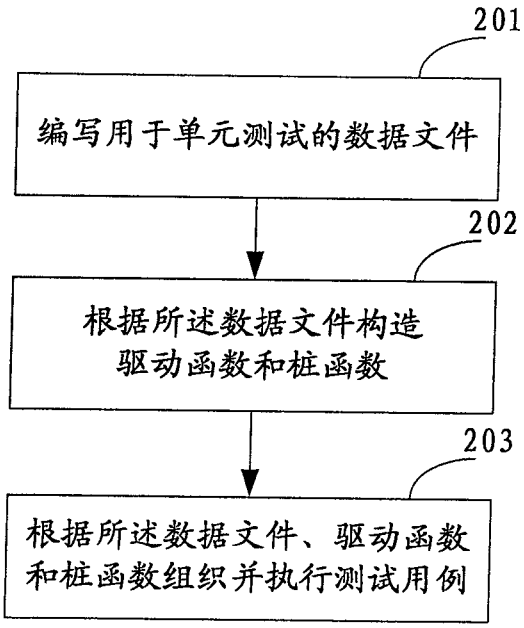

201
编写用于单元测试的数据文件
202
根据所述数据文件构造
驱动函数和桩函数
203
根据所述数据文件、驱动函数
和桩函数组织并执行测试用例

图 2

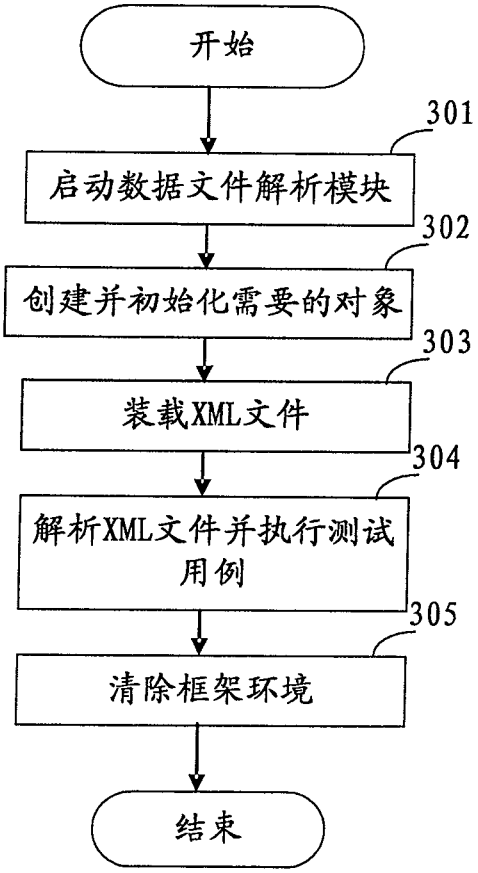

开始
301
启动数据文件解析模块
302
创建并初始化需要的对象
303
装载XML文件
304
解析XML文件并执行测试用例
305
清除框架环境
结束

图 3

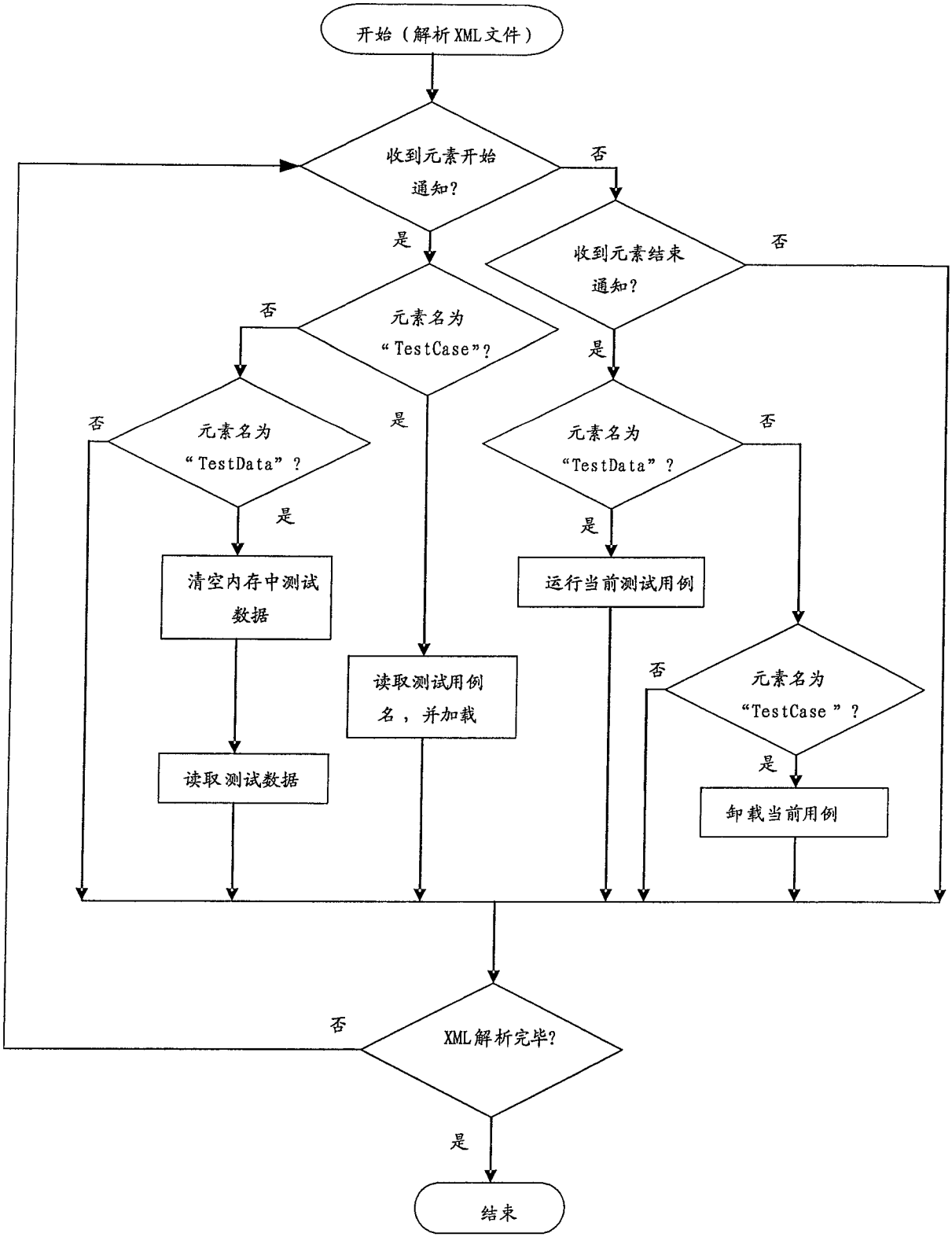

开始（解析XML文件）
收到元素开始通知?
否
是
收到元素结束通知?
否
是
元素名为“TestCase”?
否
是
元素名为“TestData”?
否
是
元素名为“TestData”?
否
是
清空内存中测试数据
读取测试用例名，并加载
运行当前测试用例
读取测试数据
元素名为“TestCase”?
否
是
卸载当前用例
XML解析完毕?
否
是
结束

图 4