# XML技术在软件可靠性测试中的应用

张 靖

(攀枝花学院网络中心 四川 攀枝花 617000)

【摘要】软件的可靠性测试是保证软件质量的重要措施。应用XML保存软件可靠性测试中生成的测试用例，提出了XML测试用例模型，实现了XML测试用例文件生成器与测试引擎，用测试驱动程序来读取测试用例，驱动被测程序运行，自动比较测试结果。能够自动、高效地对软件接口进行测试，测试数据与测试驱动程序相分离，有利于测试数据的维护与重用，在实际应用中取得了很好的效果。

**关 键 词** 软件可靠性测试； 可扩展标记语言； 文档对象模型； 测试引擎

**中图分类号** TP311 **文献标识码** A

# Application of XML Technology in Software Reliability Test

ZHANG Jing

(Campus Network Center, Panzhihua University Panzhihua Sichuan 617000)

**Abstract** Software reliability test is an important way to assure the quality of the software. This paper presents XML test case module and a way to store test cases of software reliability test by using (Extensible Markup Language) XML technology. XML test case file creator and a test engine is implemented. The test case is read by the test driver program, which drives the tested program to run and compares the test result automatically. In this way, the software interface will be tested automatically and efficiently. Test data is divorced from test driver program for maintenance and reuse of the test data.

**Key words** document object model; extensible markup language; software reliability test; test engine

软件可靠性测试是在具有软件使用代表性的环境中，获得可以用来评估软件可靠性数据的软件功能测试。例如，可以基于软件运行剖面设计软件测试用例，并用这些测试用例进行随机输入，使被测试软件运行以获得失效数据。软件可靠性测试会产生大量的测试用例，并可用于进行长时间、大规模的测试运行。测试用例属于软件测试工作的指导性文件，测试用例对测试工作的控制和指导作用相当于设计文档对编码的指导作用，尤其是在大系统中表现出对系统测试的权威性[1]。传统的测试用例一般用Word文档或Excel表的形式表示，测试人员对照测试用例逐个执行。可扩展标记语言(Extensible Markup Language，XML)技术是一项将类型和结构置于信息上层的技术。一个XML文档就是一组具有一个或者多个命名属性的结构信息项的集合。采用XML技术保存测试用例，可得到相应的XML脚本文件，既便于用户浏览，又可借助第三方工具驱动测试执行，降低了生成和维护测试脚本的难度。另外，只需配置新的解释程序，XML测试脚本便可用于不同的后端应用程序。

## 1 XML测试用例模型

XML脚本文件与通常所说的测试脚本有很大的区别[2]。通常所说的测试脚本大多用于GUI测试，主要用来记录用户与程序的交互过程，能够不断地重放，既可以手工编写，也可以录制产生。常用的脚本语言有Tcl、Phyton和Perl等[3]，测试用例主要包括用例编号、测试标题、重要级别、测试输入、操作步骤和预期结果等部分。测试脚本的生成依赖于测试输入、操作步骤和预期结果。XMl测试脚本语言主要用来描述激励函数名、参数值(输入数据)和返回值(及其结果)，而不是交互过程。利用XML Schema技术可定义XML用例模型。测试用例模型由测试组长根据被测模块编写，测试人员依照测试用

收稿日期：2007－04－18
作者简介：张 靖(1972－)，男，硕士，主要从事软件工程和计算机网络方面的研究.

例模型生成相应的测试用例，并通过XML测试用例生成向导生成XML测试文件。

利用XML Schema技术[4-5]定义的一个XML用例模型如图1所示。在图1中：

(1) TestCaseNO.表示测试用例编号，命名规则是项目名称＋测试模块名称+编号。

(2) TestTitle表示测试标题，对测试用例的用途进行描述。

(3) TestLevel表示重要级别，定义测试用例的优先级别，分为“高”和“低”两级(用户可定制自己的重要级别)。

(4) TestSite表示测试集合，包含若干个按一定顺序排列的测试步骤。

(5) TestStep表示测试步骤，对需要测试的接口进行描述，包括参数列表、接口名称和预期结果。

(6) ArgList表示参数列表，列举接口的每个参数，包括名称、注释和类型属性。

(7) InterName表示接口名称，包括名称、执行次数、出错处理属性。

(8) RValue表示预期结果，保存测试集合的预期结果。

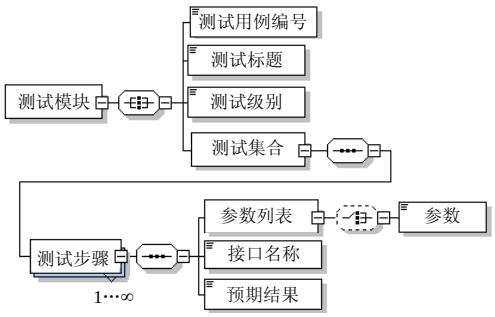


图1 测试用例XML Schema模型

## 2 XMl测试用例文件生成

### 2.1 测试用例生成

以“基于使用模型的统计测试方法”建立软件的使用模型[6]，根据马尔可夫模型产生测试用例。在马尔可夫模型中，使用模型由状态和边组成。状态表示软件使用过程中的内部环境；边表示状态间的转移关系。生成的测试用例就是从初态开始，经过若干中间状态到达终态的状态和边的序列。

用户生成测试用例序列后，利用XML测试用例生成向导得到XML测试文件。

### 2.2 XML测试用例生成向导

XML测试用例生成向导由Microsoft Visual C++和MSXML解析器组成，利用MSXML DOM解析技术动态生成XML文件(脚本)[7-8]。整个过程包括两大步骤：

(1) 读取信息的同时即对信息进行必要的封装；

(2) 根据封装的信息生成XML文件。

#### 2.2.1 信息封装

针对一个测试模块，建立三个类测试模块类Class TestModule、测试集合类Class TestSite和测试步骤类Class TestStep。

(1) 测试模块类拥有数据成员TestTitle(测试标题)、TestLevel(重要级别)和一个Class TestSite(测试集合)，以及相应的读写公共接口。

(2) 测试步骤类拥有一个Class TestStep(测试步骤)句柄容器Vector<TestStep*>、数据成员InterName(接口名称)、RValue(预期结果)和一个param数组ArgList(参数列表)，以及相应的读写公共接口。

在生成测试用例的同时，将用户输入的模块名称、测试标题、各个接口的名称、参数等信息保存在两个类的相应的数据域中，利用MSXML DOM解析技术动态生成XML文件。

#### 2.2.2 XML文件生成

MSXML是微软实现的XML解析器[5]。文档对象模型(Document Object Model，DOM)是以层次结构组织的节点或信息片断的集合，用于设计使用XML数据的、语言与平台无关的API。

一个基于树的API，可将所有数据作为节点的父子层次加载到内存中，节点可以是元素、文本、属性或其他节点类型。DOM API允许开放人员读取、创建和编辑XML数据。由于得到的测试用例呈现出树的层次结构，所以利用DOM技术可以方便地生成XML测试用例文件。为了便于使用，可对MSXML DOM创建XML文件的接口进行封装。

HRESULT CreateInstance()：创建XML文档对象

IXMLDOMElementPt CreateRoot(name, lineCover, stateCover)：创建根节点

void createElement(IXMLDOMDocument2Ptr &pXMLDom, IXMLDOMElementPtr &pe, const CString tagName)：创建元素

void createAttribute(IXMLDOMDocument2Ptr &pXMLDom, IXMLDOMElementPtr &pe, IXMLDOMAttributePtr &pa, const CString tagName, const CString tvalue)：创建属性

IXMLDOMDocumentFragmentPtr CreateFragment()：创建元素集。

在解析测试用例过程中，首先将Class TestModule的数据成员TestTitle、TestLevel生成简单元素，将TestSite生成元素集。然后遍历TestStep测试步骤句柄容器，针对每个测试步骤将InterName、RValue生成简单元素，读取param数组生成ArgList元素集。最后将得到的各个元素/元素集组成XML文档树，XML文件实例如图2所示。

```
<?xml  version="1.0" ?>
<?xml-stylesheet type= 'text/xml'  href= 'dom.xsl ' ?>
<!-- same xml file created using XML DOM object compared to cmarkup -->
<TestModule name="module1"  linecover=".24"  stateCover=".36">
<TestCaseNo.>module101</TestCaseNo.>
<TestTitle>Simplefunc</TestTitle>
 <TestLevel>Low</TestLevel>
  <TestSite>
        <TestStep>
            <InterName>f</InterName>
            <ArgList>
                <param>.33</param>
                 <param>.44</param>
            </ArgList>
            <RValue>2.2</RValue>
        </TestStep>
        <TestStep>
            <InterName>g</InterName>
        <ArgList>
            <param>4</param>
        </ArgList>
        <RValue>1</RValue>
    </TestStep>
</TestSite>
</TestModule>
```

图2　XML测试用例文件

# 3　XML测试用例文件解析

得到XML文档后，用户既可利用XSLT转换程序轻松地将其转换成HTML文档在Web上发布，又能使用测试引擎解析文档驱动测试运行。

## 3.1　解析实现

测试引擎同样采用MSXML 4.0作为XML解释器[9]，但在驱动测试运行时，使用SAX解析技术。简单XML接口(Simple API for XML，SAX)是读取和操作XML数据更快速、更轻量的方法。SAX是一个基于事件的处理器，优点是分析能够立即开始，应用程序只是在读取数据时检查数据，因此不需要将数据存储在内存中。

事实上，应用程序甚至不必解析整个文档，它可以在某个条件得到满足时停止解析。在驱动测试运行时，只需要获得测试用例中的测试步骤集信息，不需要解析整个测试用例文件。同时，不将数据存储在内存中减小了测试引擎对被测试软件运行的影响。SAX解析特点是，由事件推动应用程序(测试引擎)控制的解析过程，其解析过程如图3所示。由解析器从XML文档读取特定信息，然后解析器将内容报告给测试引擎，测试引擎推动应用程序向前运行。同时，应用程序运行时产生的事件流反馈给测试引擎以决定是否继续进行解析。可见，测试引擎是整个解析的中枢。

## 3.2　测试引擎

测试引擎由封装器(Encapsulater)和推动器(Pump)组成。

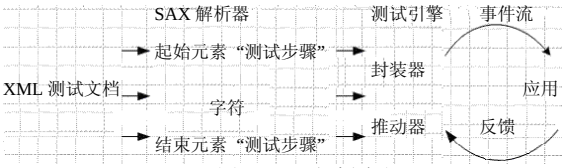


图3　SAX解析流程

### 3.2.1　封装器

为驱动被测软件运行，建立一张接口函数映射表，将被测接口与XML测试用例文档中的函数名对应起来，并建立相应的数据结构如图4所示。

```
typedef void(*PFUNC) (void);              // 标准函数指针
enum Sig                                  // 函数类型标志
{
        Sig_end = 0,
        Sig_vv,          // void (void)
        Sig_iii,         // int (int,int)
};
union PtrFunctions                        // 被测函数指针形态联合体
{
        PFUNC pfn;
        void (*pfn_vv)(void);
        int (*pfn_iii)(int,int);
};
struct Function
{
        char *nName;
        unsigned int nSig;
        PFUNC pfn;
};
Function funcs[] =                        // 函数映射表
{
        {"f",Sig_vv,(PFUNC)(void (*)(void))func1},
        {"g",Sig_iii,(PFUNC)(int (*)(int,int))func2}
};
```

图4　封装器数据结构

测试引擎运行时，获取<InterName>事件和函数名，根据函数名在函数映射表中查找参数列表标志，定义相应变量。由<param>事件将文档中的字符信息转换成相应类型参数数据，赋值给上步中定义的变量，最后由</InterName>事件调用函数。调用函数时，根据函数映射表中的nSig标志位转换联合体中的函数指针形态为对应类型。封装器同时读取<RValue>

数据保存在预期结果哈希结构中。

完成上述解析与准备步骤之后，即可驱动被测软件运行。

3.2.2 推动器

测试驱动流程如图5所示。在对XML文档内容进行重新封装后，在测试引擎中设置各种时间变量(如失效时间、失效间隔时间等)，执行状态标志(分别为通过、失败、放弃等)，并提供各种事件触发接口。初始化完成后，推动器将推动被测函数运行。推动器将应用程序反馈回的信息(实际结果)与保存在哈希结构中的预期输出进行比较，并设置执行状态标志，判断是否保存时间变量，以及是否读取下一个接口。

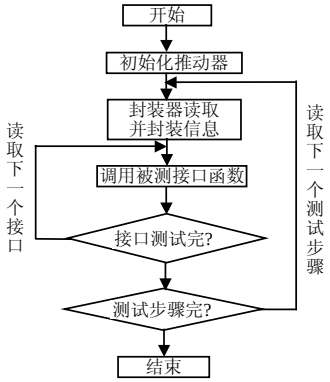


图5 测试驱动流程

## 4 结 束 语

使用XML技术保存测试用例文档既便于用户浏览，又能借助各种工具将其在Web上发布，驱动测试执行，生成测试记录，降低了生成和维护测试脚本的难度。本文着重研究了自行开发的XML测试用例文件生成器和测试引擎，以后还需要解决XML文件支持数据格式有限和测试引擎在嵌入式系统中的移植问题。

## 参 考 文 献

[1] 马瑞芳, 王会燃. 计算机软件测试方法的研究[J]. 小型微型计算机系统, 2003, 24(12): 2210-2213.

[2] 朱经纬. XML技术在软件测试自动化中的应用[J]. 计算机应用, 2005, 2(15): 94-95, 132.

[3] 王明兰, 叶东升. 测试用例描述语言研究[J]. 计算机工程与设计, 2006, 27(22): 4281-4284.

[4] 刘洪星, 张学敏, 陈 明. XML Schema设计方法研究[J]. 交通与计算机, 2006, 24(4): 123 -126.

[5] 钟玮 , 荆 涛, 吴小勇. 基于XML/Schema 的概念建模方法研究[J]. 军事运筹与系统工程, 2006, 20(1): 12-17.

[6] 冯 华, 王 戟, 徐锡山. 基于使用模型的统计测试方法的研究[J]. 计算机工程, 2002, 28(12): 93-95.

[7] 李亚辉. 基于XML描述的软件接口测试研究[D]. 西安: 西安电子科技大学, 2005.

[8] JOHNSON D J, ROSELLI P. Using XML as a flexible portable test script language[C]// AUTOTESTCON 2003. IEEE Systems Readiness Technology Conference. California: [s.n.], 2003.

[9] Microsoft Corporation. MSXML 4.0 service pack 2 (Microsoft XML core services)[EB/OL]. http://www.microsoft.com/downloads/details.aspx?FamilyID=3144b72b-b4f2-46da-b4b6-c5d7485f2b42&DisplayLang=zh-cn, 2003-05-29.

编 辑 熊思亮