基于 XML API 的组件扩展接口变异测试方法*)

聂 南[1,2] 谢晓东[1] 甘 勇[2] 卢炎生[1]
(华中科技大学计算机科学与技术学院 武汉 430074)[1]
(郑州轻工业学院计算机与通信学院 郑州 450002)[2]

摘 要 在传统的组件接口变异测试方法基础上,提出一种基于 XML API 的组件扩展接口变异测试方法。首先给出组件扩展接口测试的框架,建立起扩展接口的定义模型。XML API 在组件内部建立处理数据集的 XML 校验器,实现了原组件接口的扩展。借助组件外部的 XML Schema 变异算子,完成了组件内部数据集的自动验证和组件接口参数的测试。该方法具有多种优点,例如可视的多功能测试接口、可跨平台的通用性的测试语言等。实验表明,该方法可以应用于 COM,CORBA,EJB 等多种组件的测试环境。

关键词 组件测试,XML API,接口变异,XML Schema,变异算子

Extension Interface Mutation Testing for Component Based on XML API

NIE Nan[1,2] XIE Xiao-dong[1] GAN Yong[2] LU Yan-sheng[1]
(School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430071, China)[1]
(School of Computer and Communication, Zhengzhou University of Light Industry, Zhengzhou 450002, China)[2]

Abstract On the basis of the traditional component interface mutation, it provides an XML API-based extension interface mutation testing approach. Firstly, the framework of component extension interface testing is given, and the Definition Model towards Extension Interface (DMEI) is defined. According to XML API, XML-based data validator which extends component interface is built to process dataset inside the component. Then XML Schema mutation operator outside of component validates the component data set and tests interface functions parameters automatically. The test approach has a lot of advantages, such as the muti-functional and visual testing interface, cross-platform and general-purpose test - script language. Our preliminary experiments show that it can be applied in many component testing environments, such as CORBA, EJB and COM.

Keywords Component testing, XML API, Interface mutation, XML schema, Mutation operator

1 引言

组件技术植根于面向对象编程的原理,为软件工程开发提供了良好的技术支持。组件本身主要由三部分组成:接口、信息以及实现。对应于组件的测试,也应从接口、信息和实现三个方面加以实施。由于组件的源码不可见以及异质性,导致组件软件难以直接测试,而接口变异测试提供了一种解决途径。接口测试是白盒测试的重要一步。每个接口可能有多个输入参数,每个参数有"典型值"、"边界值"、"异常值"之分,所以输入的组合数很多。根据接口的定义,可以推断某种输入应当产生什么样的输出;输出包括函数的返回值和输出参数,如果实际输出与期望的输出不一致,那么说明程序有错误。变异测试的思想最早由 DeMillo, Lipton 和 Sayward 在 1978 年提出,它基于程序员能力假设和组合效应假设,是一种排错性测试技术。变异测试的基本原理是使用变异算子对原程序进行作用,产生大量的变异体。根据已有的测试数据运行变异体,比较变异体的运行结果和原程序的运行结果。如果两者不同,就称该测试数据将该变异体杀死。Delamaro 等人首次把程序变异用于集成测试级别[1],我们将其称为传统的接口变异测试。变异测试起初是针对白盒测试的,当它用于黑盒测试时,就称为函数调用接口的变异测试。针对不同的变异对象,还出现了规约变异与合约变异等[2]。

随着 XML 及其相关技术不断地更新,XML 数据本身的优点与其丰富的处理和验证功能,使其能够适应不同结构、不同功能软件对于数据各方面的测试要求。乔治梅森大学 Jeff OFFutt 等人研究采用基于 XML 变异分析组件交互技术生成测试用例[3]。由于测试的是异构环境下的 Web 组件,它们之间的通信与信息的交互都采用 XML 格式,Jeff OFFutt 首先提出一种交互定义模型 ISM,其由文档类型定义、消息定义和约束集合组成。测试用例是一些在 Web 组件之间传递的 XML 消息。基于 ISM 产生变异交互体和测试用例,同时对用于交互的变异操作算子进行了分类。乔治梅森大学最新发表的论文[4]表明,他们已经将 XML Schema 技术引入作为组件间消息传递的验证模型,建立了基于 XML Schema 的测试用例排错变异操作算子。

已知的工作主要致力于研究如何通过 XML 技术测试组件之间的通信,却忽略了 XML 对组件内部及其接口的测试工作。文献[5]提出了面向组件的接口变异测试方法,定义并

*)国防预研基金(51315061);中国湖北省自然科学基金(2005ABA266);河南省自然科学基金(0611054800)。聂 南 博士生,讲师,研究方向为软件工程与理论;谢晓东 博士,讲师,研究方向为软件工程与理论;甘 勇 教授,博士,研究方向为软件工程、系统集成;卢炎生 博士生导师,研究方向为软件工程与理论。

设计了 IDL(接口定义语言)描述方法的变异算子。然而,以上的研究缺乏对不同类型组件进行分析,未采用不同平台语言进行测试。本文引入作用于组件接口及内部数据集的 XML Schema 变异算子,提出一种基于 XML API 的组件接口扩展变异测试方法。该方法的测试接口与原接口彼此独立,测试框架与组件形成耦合关系。具有可视化的多功能测试接口和通用性强、可跨平台测试的语言。实验表明该方法可以实现 COM,CORBA,EJB 等多种环境下的组件测试。

2 组件接口扩展测试框架

2.1 组件扩展接口测试

本文提出的组件测试框架主要由两部分组成:

1)组件体本身基于 XML API 的数据处理模块;

2)组件外部基于 XML API 的测试模块。

为了能够校验组件中的数据集,在组件体内部引入了 XML 校验器(validator)的概念。它是一种采用 XML API 技术实现的测试数据处理器,校验器首先记录组件被调函数的输入参数和发布变量的值,将测试数据读出并且写入组件外部的 XML 的接口文档。此外,对于 XML 接口文档还可以使用 Stylus Studio 等 XML 处理工具,处理 XML 测试报告(测试结果集),实现测试接口函数及其参数值的查询。例如 XQuery 查询 XML 可以得到组件测试接口的树型结构描述。

在组件外部,主要有两个测试子模块:1) XML Schema 格式的接口验证文档处理;2) XML 接口文档处理。它们一方面可以由 XML API 自动处理,另一方面可以由 XML SPY 等工具直接处理。综上所述,给出该组件扩展接口测试的系统设计框架,如图 1 所示。

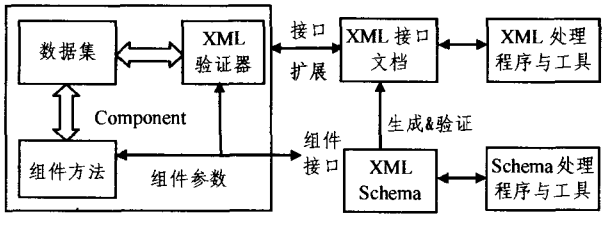

图 1 组件扩展接口测试系统框架

2.2 组件扩展接口

组件接口是组件对外的功能表现,包括组件对外的属性和函数调用,组件能够而且只能通过接口实现组件与其它组件的交互。组件接口一般又分为需求接口和提供接口两类,需求接口规定了一个组件为执行它自己的功能需要从其它组件中接受的服务,提供接口规定了一个组件提供给其它组件的服务。CORBA 组件的接口定义语言 CIDL 提供了以上服务定义,包括它的属性、函数以及接口之间的交互接口描述。本实验室 PCCM 模型研究组实现了构件模型的 CIDL 的 BNF 描述[6]与编译器开发。

由于组件主要由接口传入、传出有限的参数值,这使得接口测试的测试范围和路径很有限。已知的解决方法包括组件内建式测试,其基本思想[7]是提供者在组件内部预置测试脚本并设置相应的测试接口,使用者通过调用该接口实施组件的自行测试。

由于组件自身可划分为通用逻辑部分和可变部分,而组件的可变部分主要是属性和数据结构集。如果能够在外部对组件体的属性及数据集通过接口进行测试,则可以达到扩展测试接口的目标。由于 XML Schema 具有强大的数据描述能力,因此选择 XML Schema 文件描述组件体的属性及数据结构,通过解析 XML 文件获得组件的属性和数据源信息。

从扩展组件接口构成角度定义以下模型。

定义 1 组件扩展接口是一个二元组 $C=(D,V)$,其中:$D=(Schema^*)$是一个数据结构定义集,包含对接口格式(I)及操作方法(M)的形式定义与描述。Schema$=M^* \cup I^*$;$V=(values^*)$是一组值域的集合,*values* 是与 *Schema* 定义相对照的值的集合(也称作类型的集合),$V=S$(简单类型*)$\cup C$(复杂类型*)。

Interface(I)$=$Interface(I_F){组件功能测试}$\cup$Interface(I_S){组件结构测试}$\cup$Interface(I_C){组件执行环境配置测试}。

例如,Interface(I_F) 是一个三元组 $E=(D_F,M,V)$,其中:

$D_F=(Schema^*)$是一个数据结构定义集,包含对功能测试接口格式(I_F)的形式定义与描述。

$M=(Mi,Mo)$是功能测试接口的方法集合,Mi 表示输入方法;Mo 表示输出方法。

扩展接口 $C \in L$(L 为组件接口集),即 $L=$ {基本组件接口 L}$\cup$ {扩展接口 C}。

定义 2 规定一个接口函数有 n 个参数 $C_1,C_2,\cdots,C_n$。对于每个参数都有其指定的取值范围(约束):$C_1 \in D_1,C_2 \in D_2,\cdots,C_n \in D_n$,其中 D_n 为参数 C_n 的合理取值定义。这些参数与组件接口要处理的 XML 数据一一对应。

定义 3 D 包括 1)缺省值;2)范围值;3)c 属性:in,out,是否自动生成等;4)n 的个数,$1-n$ 顺序是可以约束的。

2.3 Schema 变异算子

从参数定义出发,建立 Schema 变异操作算子,选择测试用例来进行测试。在分析[8,9]的基础上,给出以下面向接口扩展定义的变异算子。

(1) 简单类型与面有关的变异包括:

1)简单类型正则表达式的变异;

2)界限面 minInclusive,maxExclusive;

3)数字位定义 totalDigits;

4)字符长度 String Length Change (SLC) ;

5)数值约束 Number Constraint Change (NCC)。

例如,某 password 的 Schema 正则表达式定义如下:

```
<xs:element name="password">
        <xs:simpleType>
            <xs:restriction
base="xsd:string">
            <xs:length value="6"/>
        <xs:pattern value=" ([A-Z][0-9][a-z])*"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:element>
```

简单类型变异操作包括类型、字符长度、模式改变等。

(2) 复杂数据类型定义:sequence,choice,all 三种定义的互换,实现复杂参数顺序变异,其能进行的变异操作包括顺序改变、数据个数变化等。

(3)数据类型的属性变异:例如需求(required)与禁用(prohibited)、固定值(fixed)与默认值(default)、可选(optional)等属性的互换等;此外还可以进行不同属性之间的组合变异。

对于组件体内及接口的基本数据类型以及数组、队列、hash table 等复杂的数据类型，上面定义的变异算子都可以运用，所不同的只是将变异算子作用于复杂类型中的域，即组成复杂类型的简单类型变量。下节给出具体的 XML 实例分析。

3 XML 技术实现接口测试

XML 定义的组件扩展自适应接口实现内部数据的表达、理解和提取等。

组件内部数据分为三类：

1)组件接口产生的输出值；

2)传入组件的值；

3)组件包含的要处理的数据结构集合。

为实现对这三种数据的验证，根据 XML Schema 定义，给出配置型组件的一个接口描述 XML 文档，然后通过 XML API 对它进行组件内部信息的处理。

3.1 扩展接口的 XML 表达

依照接口扩展模型建立的组件接口文档[10]定义唯一的组件函数的标识符 UUID，并将其编号作为参数而且作为其它组件的接口引用的句柄，同时也支持组件接口的检索等处理。包含功能测试接口的 XML 定义表示如下：

```
<realizes> <UUID>A.method1</UUID>//component's identifier
 <port>  <name>read_info</name>//interface name
        <parameter  name="desc" type="xxx"  />
    <parameter  name="desc2" type="xxx2" />... </port>
      <testfunction_port><function
name="insertuser"><inputs>//功能接口输入定义
<input name="userName" type="String" value=""/>
<input name="ID" type="String" value=""/>... </inputs>
      <outputs>//功能接口输出定义
<output name="status" type="boolean"/></outputs>
      </function> </testfunction_port>.....  </realizes>
```

3.2 通过 XML API 处理组件接口

测试接口的表达、组件内部 XML 校验器以及外部的 XML 处理部分都由 XML API 或者 XML 工具实现。将使用基于. NET 的 system. xml 和 JDOM[11]。例如，JDOM 能够对 XML 文档进行组件内部的函数参数初始化、传值、交换、判断等处理，还可以面向 XML schema 进行约束验证。依据本文建立的测试框架，面向该组件接口，首先由组件开发者给出一个测试规格，其中描述了组件的应用(implementation)、接口和相应的参数集(包括若干参数组)，用户可以在实际系统中使用这个规范进行测试。规范是以 XML 形式提供的，因为使用 XML 语言描述测试规格，可移植性好，可以不受操作系统的限制。系统中用到的 XML API 主要实现以下任务：

1) 组件内部数据集的提取转换为 XML 元素；

2) XML 与 XML Schema 文档的相互转换(注：system. xml 没有直接实现，补充了二次开发包)；

3) 部分 XML 与 XML Schema 文档内容的更新(变异)，其中包括数值、类型、顺序等的转换操作等；

4) XML 接口文档的检索处理；

5) 测试结果信息由 XSLT 处理表达。

依据定义的接口模型，. NET 等开发工具的 XML API 一方面获取组件内部及接口的变量值，生成测试数据；另一方面，创建组件外部的 Schema 格式的接口验证文档，然后生成 Schema 文档的变异算子，产生变异后的 Schema 格式文档；自动生成新的 XML 接口文档后，读取它们的值，重新传入组件内部，形成新的测试用例数据。综上所述，基于 XML API 的组件变异测试数据流测试图如图 2。

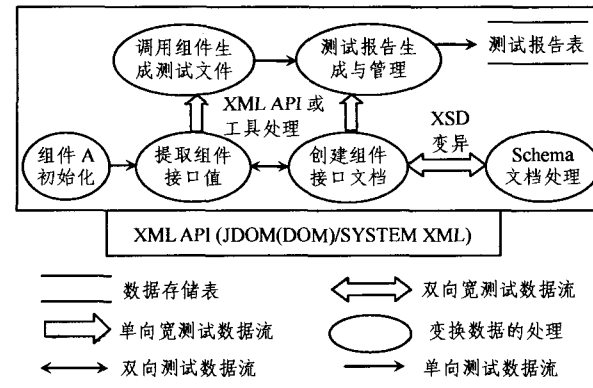

图 2 组件变异测试数据流测试

测试数据被 XML API 技术提取处理生成后，借助 Xsl API 以测试报告的形式统一管理。它需要长期不断重复记录、改进和生成，这是我们今后要继续进行的工作。

4 实验验证

为验证本文测试方法的有效性，我们在不同平台下采用不同的开发工具，以注册登录组件测试为例，进行了初步的实验分析。已知的注册登录组件包括 Microsoft 的注册登录控件 reglogincontrolsample 与 Vista 的 WinLogOn 登录过程模块，以及 IBM 开发的 WebSplitter 框架[12]中的登录组件。在此基础上，组件测试原型一方面在 Eclipse 平台下由 Java 语言开发。Eclipse 不但本身支持 XML 开发，而且 Java 也是与 XML 集成的最好的一种开发工具。目前，Eclipse 平台下已经涌现许多支持 COBRA，EJB 开发的插件，如支持 EJB 开发的 MyEclipse 或 lomboz + Jboss 等、支持 CORBA 开发的 OpenORB + ORBStudio772package. zip 等；另一方面，. NET 平台下 system. xml 是一种托管版本的 MSXML，具有更加实用的 XML 与 windows 平台组件的绑定处理功能。NET 开发的组件与 XML 及 Schema 文档格式的测试用例之间的耦合性很好，并且在组件开发及测试环境下，对测试用例进行了读、写、更新以及检索等多种自动处理。

使用以上的 XML Schema 接口模型和 XML 定义实例，本例中组件 A 提供用户注册、用户登录的安全服务。注册时包括：1)用户名称；2)密码；3)ID；4)注册时间以及其它个人信息。这些注册信息可以根据 XML Schema 定义产生，用来检查组件内部函数及接口调用的功能行为，然后通过 Schema 变异算子来产生测试用例。例如，一个程序生成的个人登录信息用例 login. xml 给出如下：

```
〈user login="2007-5-24 10:56:13"〉
  〈name〉zhangsan〈/name〉
  〈ID〉No. 1〈/ID〉
  〈password〉123456〈/password〉〈/user〉
```

XQuery 能够查询该组件测试用例中的节点数据。实现 XQuery 功能的 XQJ 是一种面向 JavaTM 的 XQuery API；. Net中支持 XQuery 功能的包括 System. Xml. Query。实现 XQuery 查询的 Stylus Studio 语句给出如下：

```
〈login〉 { for $ login in document("login. xml")//login
```

where $login/name="String"
return if ($login/time="14:20:50Z") then <test/> else $login/password }</login>

执行后得到的树型结构描述图如图 3 所示。

图 3 登录测试用例查询图

对于该用例,利用 System. xml 的 Schema 命名空间能够实现对应 Schema 文档的创建与管理;利用 XmlDataDocument 类的 dataset(数据集)等属性函数与 Schema 文档的元素进行映射,能够产生大量变异用例。通过划分等价类、边界值分析、错误推测等多种测试方法,借助开发工具创建了若干具体的变异用例,如表 1 所示。

表 1 Schema 变异测试用例种类表

简单数据变异算子/变异体		复杂数据变异算子/变异体		属性变异算子/变异体	
正则表达式数值/顺序/类型	9/6/6	简单元素顺序改变	9	可选值选择	2
数据类型个数(依照组件设计语言)	4-*			默认/固定值选择	2/2
类型长度(依照开发语言精度)	6-*	元素个数改变	6	需求/禁用值选择	2/2

注:* 表示变异体数量不定。

简单数据类型变异对应传递接口参数值组成的变化、参数类型、值大小的改变等。例如:ID 值如果类型改变,字符"1"与数字"1"的变化是否使注册信息失效;password 可以规定包括或者不包括一些符号,其类型可以是字符串,也可以是数字;如果是时间类型,可以约束时间范围:0-23:0-59:0-60Z(Z 自动生成)等。

复杂数据类型的变异主要是通过对简单类型的删除与添加等处理来实现。主要对应多个传递参数的变化,包括参数个数、顺序的改变等。例如当接口初始化时,如果缺少或者多出某些参数,将产生何种后果;如果改变类型相同的若干参数的传入次序,是否使注册信息失效等。

属性变异对应接口参数的使用约束。例如固定值属性定义常量,如果被取消后,常量就变成变量值;如果改为默认值,就改常量为初始值。通过变异体的变异,可以测试 XML 接口多种初始化值,观察传递不同约束下组件注册或登录值的执行后果。

结束语 本文提出组件外部 XML Schema 与内部 XML 校验器内外结合测试组件的方法。通过 XML API 实现原组件接口的扩展,外部建立 Schema 定义的测试脚本数据模型,引入作用于接口的 XML Schema 变异算子,实现了组件外部的 Schema 算子对于组件接口的自动验证与测试。它建立起可视化的多功能测试扩展接口,使用通用性的 XML 格式的测试脚本。这种做法,使得测试脚本和组件的耦合度变得既松散又统一,降低了生成和维护测试脚本的难度。由于 XML 已经成为多种语言环境的数据描述格式,XML API 已经被多种语言采用,因此该方法可以应用于 CORBA,EJB,COM 等多种组件的测试环境,从而具有跨平台性等多种优点。不足之处在于对于该方法的变异覆盖准则缺乏有效分析,需要对更多的组件进行测试和统计;建立的组件扩展接口对组件间的通信支持不够,需要建立实现接口转换的适配器并且进行相应的变异测试。此外,基于 XACML 的面向组件访问控制(access control)的变异测试也是我们下一步需要研究的内容。

参 考 文 献

[1] Delamaro M E, Maldonado J C, Mathur A P. Integration testing using interface mutations// Proceedings of International Symposium on Software Reliability Engineering (ISSRE'96). [s. l.]: IEEE Computer Society Press, 1996: 112-121

[2] 姜瑛,辛国茂,单锦辉,等. 一种 Web 服务的测试数据自动生成方法. 计算机学报,2005,28 (4) : 568-577

[3] Lee S C, Offutt J. Generating test cases for XML-based web component interactions using mutation analysis// Proceedings of the International Symposium on Software Reliability Engineering. 2001: 200-209

[4] Xu Wuzhi, Offutt J, Luo Juan. Testing web services by XML perturbation// Proceedings the 16th IEEE International Symposium on Software Reliability Engineering. 2005: 257-266

[5] 杨建军,陈卫东,叶澄清,等. 面向组件的接口变异测试方法. 浙江大学工学版,2003(372): 129-133

[6] 卢炎生,查虎平,徐丽萍. PCCM:具有性能约束的组件模型. 计算机科学,2004,31(5): 89-92

[7] Beydeda S. Research in testing COTS components built in testing approaches// Proc. of the 3rd ACSPIEEE Int'l Conf. on Computer Systems and Applications. Los Alamitos, CA: IEEE Computer Society Press, 2005: 101-104

[8] Li Jian - Bing , Miller J . Testing the Semantics of W 3 C XML Schema// Proceedings of the 29th Annual International Computer Software and Applications Conference, COMPSAC'05. Volume 1. July 2005: 443-448

[9] Walmsley P. XML 模式权威教程. 北京:清华大学出版社,2003

[10] Chen Huoping, Hariri S, Rasul F. An Innovative Self-configuration Approach for Networked Systems and Applications// IEEE International Conference on Computer Systems and Applications, 2006: 537-544

[11] JDOM. JDOM v1. 0 API specification. 2006. http://www.jdom.org/docs/apidocs

[12] Han R, Perre V. Nagshineh M. WebSplitter: A unified XML framework for multi-device collaborative web browsing// Proceedings of the ACM Conference on Computer Supported Cooperative Work. 2000: 221-230