第41卷 第12期 西 安 交 通 大 学 学 报 Vol.41 №12
2007年12月 JOURNAL OF XI′AN JIAOTONG UNIVERSITY Dec. 2007

软件集成测试中的一种用例生成方法

王战敏[1]，崔杜武[1]，费蓉[1]，张淳民[2]
(1.西安理工大学计算机科学与工程学院，710048，西安；2.西安交通大学理学院，710049，西安)

摘要：针对面向对象语言的多态、动态绑定等特性增加了面向对象软件集成测试难度的问题，提出了一种测试用例的生成方法.首先借鉴正交矩阵测试策略的思想，采用自定义正交矩阵生成算法生成一个二维正交矩阵，再使用鲁棒性测试方法优化生成正交矩阵，最后采用自定义测试用例生成算法为面向对象软件的集成测试生成测试用例集，并将测试用例集用XML文档保存，以备下一步测试用例复用.经验证表明，使用正交矩阵能提高错误检测能力，用其生成的测试用例比较少且方法简单、易于实现.

关键词：面向对象软件；集成测试；正交矩阵测试；用例生成方法
中图分类号：TP311 **文献标识码**：A **文章编号**：0253-987X(2007)12-1427-04

Method for Generating Test Cases in Software Integration Testing

Wang Zhanmin[1], Cui Duwu[1], Fei Rong[1], Zhang Chunmin[2]
(1. School of Computer Science and Engineering, Xi′an University of Technology, Xi′an 710048, China; 2. School of Sciences, Xi′an Jiaotong University, Xi′an 710049, China)

Abstract: Focusing on the problem that the features of multimode and dynamic binding of object-oriented software increase the difficulty of integration test, a novel method for generating test cases is presented. Firstly benefiting from the idea of OATS (orthogonal matrix testing strategy), a 2D orthogonal matrix is created according to self-defined orthogonal matrix generation algorithm. Then the generated orthogonal matrix is optimized by using robustness test. Finally, the testing case set is created for the object-oriented software integration test using the self-defined test cases generation algorithm, and the cases sets are saved with XML documents for re-using in next step testing. The experiments show that the ability of error detection is improved with orthogonal matrix, the generated cases are less, and the proposed method is simple and easy to be realized.

Keywords: object-oriented software; integration testing; orthogonal matrix testing; case generation method

面向对象软件测试可分为4个级别：算术级(方法级)、类级、集成测试级、系统级[1]，其中集成测试作为重要环节，贯穿于面向对象软件构造过程的始终.

在面向对象软件集成测试中，最需要克服的问题是类间交互带来的多态和动态绑定问题，目前国内外有关动态绑定的研究都没有提出一个彻底的解决方法，这一级测试目前常用的方法有：基于合同规约的方法[2]、基于状态的方法、确定性测试方法、数据流分析方法[3]以及TACCLE[4]方法等，它们均存在一定的局限性.

面向对象软件测试的核心问题是生成测试用例.测试用例是软件测试必须遵守的准则，是软件测试质量的根本保障.

收稿日期：2007-02-08. 作者简介：王战敏(1969～)，女，在职博士生，讲师. 基金项目：国家自然科学基金重点资助项目(40537031)；国家自然科学基金资助项目(40375010，60278019).

本文在完全覆盖测试不可能实现的情况下，提出了一种面向对象软件集成测试用例生成方法.

1 类间测试影响因素

进行类间测试，必须明确需要检测的错误类型. Leung 和 White 提出了一种集成测试模型[5]，并认为面向对象软件集成测试主要是测试接口和功能 2 类错误. 无论测试哪一种错误，测试用例都是其中的关键问题.

本文假设如下.

(1)一个类的定义都是规范的，可以从中得到类的正确规范、类的继承关系以及每一个对象的可能状态.

(2)若已经成功完成了类级测试，那么每一类以及类的内部方法之间的交互都是正确的.

(3)本文仅讨论 2 个类之间的交互测试，而且这 2 个类都拥有子类.

(4)对于全局的值和参数以同样的方法处理.

对象是程序运行的实体，对象封装行为也封装状态. 对象的状态对类间测试有很重要的作用，而且类的状态测试是面向对象软件测试的重要部分. 在类级测试成功完成的基础上，再进行集成级测试时，对于类的状态信息的提取可以直接采用上一级的测试结果，从而减少工作量.

在本文中，“状态”指的是 McGregor 和 Dyer[6] 描述的状态设计级别. 对于类的继承情况，如果父类和子类的状态是一致的，则无需重新测试，否则应该测试子类.

2 类间多态性测试

多态性指表达式、函数、过程及其操作对象可以有一种以上的不同类型，分为多态操作和多态变量. 操作的多态性指在类继承等级的不同层次中可以是一个共享方法，而不同层次中的类可按各自的需求调用该方法. 多态变量指该变量可引用不同类的对象. 一个多态变量的静态类型是在程序中声明的类型，而动态类型则是程序运行时实际引用的对象类型. 如果父类与子类的关系同时体现了类型与子类型的关系，则继承将包含多态.

2.1 正交矩阵测试策略

Mandl[7]将正交拉丁方阵应用于编译器测试，基于此本文在实验中将正交拉丁矩阵引用到软件测试中，利用生成的正交矩阵转换生成测试用例，这种方法能有效减少测试用例个数. 这样的一个正交矩阵必须具有如下 5 个因素.

(1)行：矩阵中行的个数. 它应对应于使用正交矩阵测试策略(OATS)产生的测试用例个数.

(2)因子：矩阵中列的个数. 它应对应于矩阵中可以控制的变量最大个数.

(3)级别：单个因子值的最大个数. 一个正交矩阵中包含的级别 N 的值可用 $0\sim N-1$ 表示.

(4)强度：列中级别强度可以用级别出现的频率来表示.

正交矩阵有固定的表示方法，如 $\boldsymbol{L}_9(3^4)$ 表示这个正交矩阵有 9 行，其中包含 4 个因子和 3 个级别强度为 2 的级别. 假定测试中有 4 个类，每个类有 3 个状态，根据排列组合则有 81 个测试用例，其中有很多测试用例是无效的，这样会增加测试成本，降低测试效率.

虽然可以将 OATS 直接引入软件的集成测试之中，且能够显著减少测试用例的个数，但对于面向对象软件的集成测试，OATS 生成的测试用例集所给出的测试用例个数不能实现充分测试. McGregor[8] 提出了面向对象的类交互测试 OATS 测试方法，该方法在测试过程中存在类状态盲目组合，从而导致：①测试用例爆炸；②矩阵不易扩充. 当一个类族的层次进一步增加时，原有的正交阵列不再适用，需要重新选择一个大矩阵，这将使整个交互测试过程重复进行，因此需要改进 OATS.

2.2 改进的 OATS

利用正交矩阵生成的测试用例能覆盖每一个因子的值，但并不能覆盖所有值的组合，因此以正交矩阵产生的测试用例集是最小的. 对于此问题，借鉴鲁棒性测试[7,9-10]可以为测试用例定义一个可选的过程，以此解决多态性和对象状态问题. 根据测试的意图可以适当地增加测试用例的个数，提高覆盖率，但是前提是需要明确预期的测试结果.

无论采用标准正交矩阵还是普通正交矩阵，在面向对象软件的集成测试中，发送对象、发送对象的状态、接收对象、接收对象的状态以及对象的参数、对象参数的状态均会影响 2 个对象交互信息，所以在进行类间测试时，必须考虑以上 6 个因素.

利用 OATS 可以生成矩阵，其中假定客户类 $C=0,1,2$，服务器类 $S=0,1,2$，消息数 $M=0,1,2$，则加入匹配因子之后的正交矩阵如表 1 所示.

2.3 正交矩阵生成算法

在正交矩阵生成算法(CREATE_OA)中，发送类、接收类和参数类是输入的参数，程序的返回值是

表 1　正交矩阵表

测试用例	C	S	M
TestCase1	0	0	0
TestCase2	0	1	1
TestCase3	0	2	2
Test4Case	1	0	1
TestCase5	1	0	2
TestCase6	1	1	2
TestCase7	2	0	2
TestCase8	2	1	0
TestCase9	2	2	1

一个指向正交矩阵的指针(OA-REFERANCE).

CREATE_OA 在实现的过程中,首先计算因子数、级别数、自由度等级. 正交矩阵中的行就是测试用例的个数,矩阵中的列就是因子的大小,因此可以采用一般的方法生成正交矩阵,如标准的或普通的正交矩阵. 在实现的过程中,采用两重 for 语句循环生成一个二维正交矩阵,最后返回一个指向矩阵的指针. CREATE_OA 步骤如下.

步骤 1: 将类列表中的每一个类作为发送类,并且计算类所具有的可替换的多态子集.

步骤 2: 发送类中的每一个方法以及每一个类之间传递的消息将做如下工作:确定接收类、参数类,计算类所具有的多态子集集合. 如果发送类、接受类或者参数类没有多态子集,就直接构建测试用例,并直接写入 XML 文档,同时返回测试用例构建消息;反之,在程序结束后返回一个指向正交矩阵的指针.

3　测试用例的生成

类间测试用例的构造方法调用了 2.3 节中的 CREATE_OA 算法,测试用例生成算法(CREATE_CASE)的伪码如图 1 所示.

在 CREATE_CASE 的实现过程中,利用了自定义的 TRAN-ARR()函数生成测试用例阵列,然后为测试用例阵列中的每一个测试用例构建消息.

一般来讲,信息量越丰富,生成和维护的代价就越高,管理的繁琐程度就越大,测试用例的复用程度也就越高.

XML 允许用户定义无穷个标记来描述文件中的任何数据元素,将测试用例保存到 XML 文档中,

```
Function CREATE_CASE(ClassList: )
{
    OA-REF=CREATE_OA(sender, receiver, list of parameters)
    TestCaseArray=TRAN-ARR (list of classes , OA-REF)
  For each test case in TestCaseArray{
    Construct test message and write it into the XML Document
by given format and check the test case avoiding reiteration
}
TRAN_ARR()函数的实现如下:
Function TRAN_ARR (list of classes ,OA-REFERANCE)
{ For every class of ClassList
   { For the first colum of OA which is SenderClass
   {read information of senderclass , recervier class and the
   parameter lists
        write these informaton into a TestCaseArray
    If add the testcase then
     Input the classes that used more and input the states of classes
     that used more
    write these testcase into theTestCaseArray
    return TestCaseArray
}
```

图 1　测试用例生成算法的伪码

能够将文档的内容组织成丰富、复杂的完整信息体系,使得各种类型的应用程序和设备更易使用、存储、传送和显示. XML 语言具有自描述性、良好的数据存储格式和可扩展性、高度的结构化以及内容和形式分离等特点,为以后进行用例的复用提供方便.

因此,可将消息按规定的格式写入 XML 文档中,其中包括测试用例序号,发送类的级别、名称、状态,接收类的级别、名称、状态,参数类的级别、名称、状态. 在测试用例生成过程中,会自动过滤掉重复的测试用例,然后以 XML 格式存储用例. 用例的 XML 文档的描述如图 2 所示.

CREATE_CASE 算法在程序中调用了传输矩阵函数 $\boldsymbol{F}_{\text{TRAN-ARR}}$,该函数与类的列表和 OA-REFERANCE 有关. 在 CREATE_CASE 算法中至少涉及到 4 个因子,分别代表发送类、发送类的状态、接收类、接收类的状态. 计算因子的级别数是其中的一个关键步骤,每一个对象因子的级别数应与程序运行时能够表示此类的类数量相等.

先计算自由度等级、因子的数目以及级别,再将这 3 个值与标准正交矩阵的相似值进行比较,如果比标准正交矩阵的因子和级别的数量小,那么生成并存储一个标准正交矩阵,反之生成、存储一个普通的正交矩阵.

对于特殊情况,需要使用虚级别去补充标准正交矩阵的维数.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified" attributeFormDefault
="unqualified">
<xs:element name="TestCase">
    <xs:annotation>
    <xs:documentation>comment describing your root element
</xs:documentation>
    </xs:annotation>
    <xs:complexType>
    <xs:sequence>
    <xs:element name="TestCaseNum"/>
     <xs:element name="SenderClassName"/>
     <xs:element name="SenderClassLevel"/>
     <xs:element name="SenderClassState"/>
 <xs:element name="ReceiverClassName"/>
 <xs:element name="ReceiverClassLever"/>
 <xs:element name="ReceiverClassState"/>
 <xs:element name="ParameterClassName"/>
 <xs:element name="ParameterClassLever"/>
 <xs:element name="ParameterClassState"/>
    </xs:sequence>
    </xs:complexType>
    </xs:element>
    </xs:schema>
```

图2 用例的XML文档描述

针对表1中的例子，利用本文方法，并根据测试的意图以及类级的测试结果可生成如表2所示的测试用例.

表2 测试用例表

TC	SN	SL	SS	RN	RL	RS	MN	ML	MS
1	*C*	0	0	*S*	0	1	*M*	0	1
2	*C*	0	1	*S*	1	2	*M*	1	2
3	*C*	0	2	*S*	2	0	*M*	2	1
4	*C*	1	0	*S*	0	0	*M*	1	1
5	*C*	1	1	*S*	0	1	*M*	2	1
6	*C*	1	2	*S*	1	1	*M*	2	2
7	*C*	2	0	*S*	0	0	*M*	2	0
8	*C*	2	1	*S*	1	1	*M*	0	0
9	*C*	2	2	*S*	2	2	*M*	1	1
10	*C*	0	2	*S*	1	1	*M*	2	0
11	*C*	0	1	*S*	2	2	*M*	1	1
12	*C*	1	1	*S*	2	2	*M*	1	0
13	*C*	1	1	*S*	2	2	*M*	2	0
14	*C*	2	2	*S*	1	1	*M*	2	1

TC：测试用例序号；SL：发送类的级别；SN：发送类的名称；ST：发送类的状态；RL：接收类的级别；RN：接收类的名称；RS：接收类的状态；ML：参数类级别；MN：参数类名称；MS：参数类状态.

生成的测试用例除了与利用OATS生成的正交矩阵有关以外，还与测试者的意图以及类级的测试结果相关. 对于相对重要的类及类的某些相对重要的状态，可以增加该类以及类状态出现的次数，增加测试用例，以弥补OATS的不足.

4 结 论

针对面向对象软件集成测试用例生成法进行了探讨，目的是解决面向对象软件的多态性问题，并通过多态定位逐步解决动态绑定问题.

在测试用例中，本文使用正交矩阵可提高错误检测能力，有效减少测试用例的数量，且方法简单、易于实现，并为测试用例复用提供了方便. 不足的是，对测试人员提出了较高的要求，且并不能完全解决动态绑定问题，集合考虑的范围局限于多态替换和状态代表的测试用例集，对于穷尽测试检错率，有待进一步研究.

参考文献：

[1] Smith M D, Robson D J. A framework for testing object-oriented programs [J]. Journal of Object-Oriented Programming, 1992, 5(3): 45-53.

[2] 孙玉霞，陈火炎. 面向对象软件簇级的一种动态测试工具的设计与实现 [J]. 小型微型计算机系统，2003，24(3)：376-379.
Sun Yuxia, Chen Huoyan. Design and implementation of a dynamic testing tool for object-oriented software at cluster level [J]. Mini-micro Systems, 2003, 24(3): 376-379.

[3] Chen Wing-Kwong, Chen Tsong-Yueh, Tse T H. An overview of integration testing techniques for object-oriented programs[C]// Proceedings of the 2nd ACIS Annual International Conference on Computer and Information Science. Pleasant, USA: International Association for Computer and Information Science, 2002: 116-122.

[4] Chen Huoyan, Tse T H, Chen Tsong-Yueh. TACCLE: a methodology for object-oriented software testing at the class and cluster levels [J]. ACM Transactions on Software Engineering and Methodology, 2001, 10(1): 56-109.

[5] Leung H K N, White L. A study of integration testing and software regression at the integration level [C]// Proceedings of Conference on Software Maintenance. Los Alamitos, USA: IEEE Computer Society, 1990: 290-301.

[6] McGregor J D, Dyer D M. A note on inheritance and state machines [J]. Software Engineering Notes, 1993, 18(4): 61-69.

[7] Mandl R. Orthogonal latin squares: an application of experiment design to compiler testing [J]. Communications of the ACM, 1985, 28(10): 1054-1058.

[8] McGregor J D, Sykes D A. 面向对象的软件测试 [M]. 杨文宏，李新辉，杨洁，译. 北京：机械出版社，2002.

[9] Phadke M S. Quality engineering using robust design [M]. Englewood Cliffs, USA: Prentice Hall, 1989.

[10] Raghavarao D. Constructions and combinatorial problems in design of experiments [M]. New York: John Wiley & Sons Inc., 1971.

（编辑　苗凌）