

松散耦合接口测试数据精简技术

李 越

(江苏自动化研究所,江苏连云港 222006)

摘 要:软件接口测试的重点是要检查数据的交换、传递和控制过程,还包括处理的次数。在接口测试中经常涉及大量具有复杂关系的接口数据,这些接口数据形成的输入集合相当巨大,对软件测试人员造成很大的困扰。该文研究的是基于松散耦合的接口,如何在巨大的输入数据集合中确定有效的测试用例输入子集。提出了通过精简输入数据域、确定值域相关关系以及多重维数相邻因素组合覆盖表等技术设计接口测试用例的方法。应用该方法,在限定资源条件下,简化输入集合,可以得到更有效的测试数据。

关键词:软件测试;接口测试;松散耦合

中图分类号:TP311.5

文献标识码:A

文章编号:1673-629X(2011)06-0223-04

Research on Condense Input Data Based on Loose Coupling Data Interface

LI Yue

(Jiangsu Automation Research Institute, Lianyungang 222006, China)

Abstract: It is important that checking the data exchange and transfer, controlling the process and times for software interface testing. It makes a big problem for software testers that there are massive complicated interface values for software testing. This paper present a method based on loose coupling data which can provide for effective data by reducing input data, confirm correlativity of the input field and the covering array generation algorithms for variable strength neighbor factors coverage. By using this method, it can get more effective testing data, for reducing input set in a limit resource condition.

Key words: software testing; interface testing; loose coupling

0 引 言

接口测试是软件测试的重要组成部分,作为测试输入的数据不但要包含正常数据,还要包含边界值、异常值及数据间的组合关系等,以验证接口关系的正确性和健壮性。在这些交互接口值域关系中,又以数据间组合关系最为复杂,是测试的重点和难点。樊玮等^[1]提出使用进化测试的方法,孙淑香等^[2]提出基于树型模型的黑盒测试用例自动生成法,均描述了关于测试用例的精简技术。考虑到数据间组合关系复杂性,该文讨论对具有松散耦合性质的接口,如何快速生成数量少、质量高的测试用例集,以提高效率,降低成本。

1 术 语

1.1 松散耦合

耦合性是程序结构中各个模块间相互关联的度

量。如果一个模块访问另一个模块时,彼此之间是通过数据参数(不是控制参数、公共数据结构或外部变量)来交互,则这种关系称为数据耦合,由于限制了模块间只能通过数据表进行交互,所以,数据耦合是松散耦合。引申定义独立软件间的相互关联为接口间的耦合。该文研究的对象是基于松散耦合的接口。

1.2 接口数据相关性

接口数据相关性,是指每个接口中各数据域之间的相关性。从关系角度看这种接口数据的相关性由两部分组成:值域的约束性和值域的相关性。其中,值域约束性是指接口数据项自身具有取值范围,并且对取值范围有特定合法数据子集的要求;值域相关性指的是某一数据项的值域和同一接口中另一数据项的值域之间存在的一种或多种相互约束。

2 问题的提出

2.1 测试用例集

在接口数据相关性中,值域约束性存在如下经验^[1]:接口在处理合法的数据输入值时,出现错误的概

收稿日期:2010-10-30;修回日期:2011-01-29

基金项目:总装备部“十一五”预先研究项目(51319070102)

作者简介:李 越(1979-),黑龙江安达人,工程师,研究方向为软件测试。

率远小于对非法数据输入的处理。所以,接口中任何一个数据域,其测试用例集均需包含 3 部分:合法数据子集、非法数据子集以及取值范围非法的数据子集。

针对不同接口数据域存在如下测试生成方法:

1) 常规测试用例生成方法:根据接口数据域的变量类型采用边界值分析的方法。如对于字符型数据域,可以考察字符表的边界字符以及字符表之外的非字符输入(即非法数据类型),而对于整数型的数据域,考察变量所能代表的最大、最小整数以及数据溢出等非法类型。

2) 带约束条件的测试用例生成方法:有些接口数据域,有其特定的约束条件限制,比如:不同年月日的限制、正常气温的限制范围等。这些都会影响接口测试域的取值范围,不能单纯只考虑常规测试用例的生成方法。

3) 接口数据域为数组的测试用例生成方法:由于数组越界是导致内存出错的最常见问题,所以一定要考虑:长度非法和长度合法,其中长度非法作为重点,因为数组的越界溢出是交互接口的最大考验,一般采用长度过长或过短,至于每一项的取值则随机生成。

2.2 接口测试用例集

设一待测系统 SUT 的交互接口数据域为: $I = \{c_1, c_2, \dots, c_n\}$, $c_i (1 \leq i \leq n, n \text{ 为数据域个数})$ 为用户规定顺序的第 i 个数据域。测试用例集为: $P = \{P_1, P_2, \dots, P_n\}$, 数据域 c_i 的测试用例集为 P_i 。设 P_i 为 $\{v_{i1}, v_{i2}, \dots, v_{ik}\}$, 其中 $k = |P_i|$ 。

定义 1 称 n 元组 (v_1, v_2, \dots, v_n) , 待测系统 SUT 的测试用例即为 v_i , 其中 $(v_1 \in T_1, v_2 \in T_2, \dots, v_n \in T_n)$ 。

定义 2 对于待测系统 SUT, 取 $k (1 \leq k \leq n)$ 个参数值后形成 n 元组 $\text{mod} = (-, \dots, v_{1k}, -, v_{2k}, \dots, v_{nk}, -, \dots, -)$, 其中“-”表示该位置参数值未确定。称 mod 为待测系统 SUT 的一个 k 值模式, 简称模式。当 $k = n$ 时, 对应模式称为全模式, 即为 SUT 的一个测试用例。

推论 1 设 S 为待测系统 SUT 全模式 mod 的集合, 则 S 为测试用例集的笛卡尔乘积, 即

$$S = P_1 \times P_2 \times \dots \times P_i \times \dots \times P_n$$

$$= \{ \langle v_{11}, v_{21}, v_{31}, \dots, v_{n1} \rangle, \langle v_{12}, v_{21}, v_{31}, \dots, v_{n1} \rangle, \dots, \langle v_{1k1}, v_{21}, v_{31}, \dots, v_{n1} \rangle, \langle v_{11}, v_{22}, v_{31}, \dots, v_{n1} \rangle, \dots, \langle v_{1k1}, v_{2k2}, v_{3k3}, \dots, v_{iki}, \dots, v_{nkn} \rangle \} \quad (1 \leq i \leq n, n \text{ 为数据域个数}, v_i \text{ 为第 } i \text{ 个域的一个输入}, k_i \text{ 为各个 } P_i \text{ 离散点集的大小})$$

由推论 1 可知, 即使将每个数据元素的输入个数定义在有限的范围 P 内, 由于笛卡尔乘积的特性, 得到的接口测试用例集仍然非常庞大, 如果照搬执行, 将耗费大量的时间和精力。

3 解决方法

3.1 精简数据域

在规模庞大的系统中, 有时为了简约接口关系, 多个设备软件的输入数据遵循相同的接口协议, 这就导致了对于某一独立的软件系统 SUT 而言, 它所需要的接口数据域 P 仅仅是接收外部接口数据 I 的子集, 即 $P \subset I$ 。如果能够在进行测试用例集的设计之前, 将 I 精简, 确定 P 的集合范围, 将对减少接口测试用例的个数有较大的帮助。

当然, 可以通过软件研发人员了解哪些接口数据是有用的。但是, 这就象询问软件是否存在未使用的变量一样, 软件的设计与实现通常并不能完全统一, 必须要由软件测试人员自己来确定数据的有用性。

设计算法如下:

算法 1:

输入: 接口数据域 $I = \{c_1, c_2, \dots, c_n\}$, $c_i (1 \leq i \leq n, n \text{ 为数据域个数})$ 。

输出: 接口程序对数据域 I 中各个参数的复杂性集 $M = \{m_1, m_2, \dots, m_n\}$, $m_i (1 \leq i \leq n, n \text{ 为数据域个数}, m_i \text{ 为各个参数在接口程序中的复杂程度, 初始值为 } 0)$ 。

步骤:

- 1) for $i = 1$ to n ; {
- 深度优先搜索程序树, 直到遍历程序 {
- 2) if (存在 $c_i(c_i')$ 的传值操作) then ; { 将被赋值域记为 c_i' ; 继续当前子树的深度遍历; }
- 3) else if (存在 $c_i(c_i')$ 的判断操作;) then ; { 返回父节点, 进行下一子树深度遍历; }
- 4) else { 合并该语句节点; }
- 5) 计算产生的流程图圈复杂度: $m_i = E - N + 2$; $\{ (E \text{ 是流图中边的数量}, N \text{ 是流图中节点的数量}) \}$
- 对于程序 fun ()

```
//计算长方体的底面积和体积
int fun(float x, float y, float z, .....u.....)
{
    float area, volume;
    int value;
    value = 0;
    //计算面积
    area = x * y;
    //计算体积
    volume = x * y * z;
    if (area > 1)
    {
        value = value + 1;
    }
    else
    {
        value = value - 1;
    }
    if (volume > 1)
    {
        value = value + 1;
    }
    else
    {
        value = value - 1;
    }
    return (value);
}
```

应用算法1,参数 x, y 的流程图如图1所示。

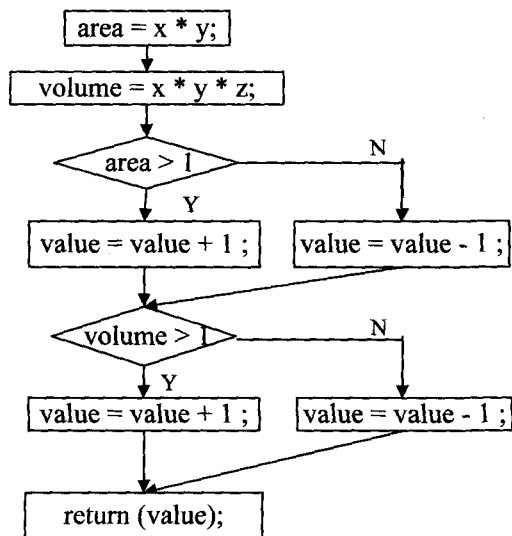


图1 参数 x, y 的流程图

由上图可知 x, y 的圈复杂度为3。

参数 z 的流程图如图2所示。

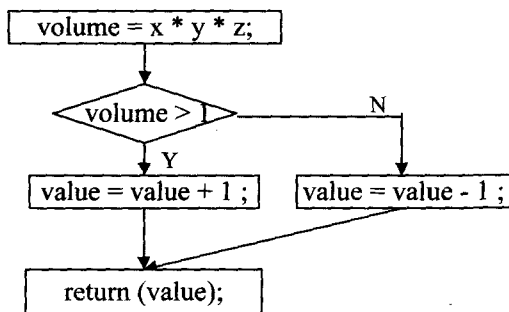


图2 参数 z 的流程图

由上图可知 z 的圈复杂度为2。

对于参数 u ,圈复杂度为0。

得到所有 $mi=0$ 的数据域 Q ,则 $P=\bar{Q}(P \subset I, Q \subset I)$ 。

现在,集合 S 可以进行精简,即 $S=T_1 \times T_2 \times \dots, T_i \times \dots, T_p(1 \leq i \leq p, p \text{ 为数据域 } P \text{ 的个数}, p \leq n)$ 。

3.2 确定值域相关关系

即便已经得到精简的数据域,接口测试用例集 S 仍然规模庞大,而组成 S 的笛卡尔乘积表明全部数据域均具有相关性,为了减少无效的测试用例,需要确定重点测试对象并缩减数据域之间的相关性。

1) 重点数据域。数据交互接口中,并非每个数据域都有相同的重要性,对于像速度、数值、字符、位置、地址、尺寸等有数量意义的数据域,可以作为重点数据域,着重进行考虑。

2) 测试项之间的各种关系。很多情况下,系统的问题起源于接口数据域内域之间的各种关系。针对数据域之间的这些关系(如依赖关系、大小关系、函数关系等),可以采用不同的测试用例。例如,两个域 x, y

之间存在着包含关系($x \in y$),对于 x 取值范围在 y 之外的情况违反了这种规律,应该作为必要的测试用例。

3.3 多重维数相邻因素组合覆盖表

确定值域相关关系后,采用组合测试的方式进行测试用例的输入设计。自从2002年以来,Reilly等研究了组合测试的可用性,发现两两组合测试具有其实际应用价值^[3],之后,Kobayashi和Tsuchiya对两两组合提出了初步算法^[4],William对两两组合的应用场景进行了深入研究^[5],Cohen M. B.和他的同事研究了运算效率和运算结果之间的关系,得到更优化的算法^[6-8],Wallace和Kuhn对组合应用到复杂系统进行了深入研究^[9],Tsuchiya和Shiba等人研究了蚁群算法和遗传算法在两两组合自动化测试中的应用^[10]。Colbourn等人提出了将确定性密度算法应用在两两组合测试中^[11]。Schroeder等人通过实验比较了随机测试与同规模组合测试的错误检测能力^[12]。Colbourn和Sherwood等人进一步对其各种两两组合的形态进行研究,并取得了一定的进展^[4-14]。但是在实际的软件系统中,一般不存在任意两个因素之间都存在相互作用的情况,大多数情况下,一部分因素之间不存任何关系,而另一些因素之间则存在较强的相互作用,这其中,我们考察一种比较特殊的情况,那就是只在相邻因素之间存在较强的相互作用,王子元等^[15]对参数集合 F 的子集 $F_i(i=1, 2, \dots, t)$ 提出了多重维数相邻因素组合覆盖表生成算法。

定义3 设 $A=(a_{i,j})_{m \times n}$ 为一个 $m \times n$ 的矩阵,其中第 j 列中所有元素均取自集合 $V_j(j=1, 2, \dots, n)$ (j 列表示SUT的参数 f_j),即 $a_{i,j} \in V_j$ 。给定正整数 $N(N \geq 2)$,如果 A 中任意 N 列(设为第 i_1, i_2, \dots, i_N 列)均满足:所有的 N 维组合 $(V_{i_1}, V_{i_2}, \dots, V_{i_N})$ 均在这 N 列的 N 元有序组中以相同的频率出现,则称 A 为正交表,设为 $OA(m; N, F)$;如果不要求等频率,但每一个组合至少出现一次则称 A 为 N 维组合覆盖表,设为 $CA(m; N, F)$;如果 A 中任意相邻的 N 列均满足:在这 N 列所形成的 N 元有序组中, $V_{i_1}, \dots, V_{i_{i+N-1}}$ 中所有的 N 维组合至少出现一次则称 A 为相邻因素 N 维组合覆盖表,设为 $NCA(m; N, F)$ 。 A 中,行表示测试用例 m 代表测试用例的数量如果 m 是能够保证 N 维组合覆盖表 A 成立的最小正整数则称 A 是最优相邻因素 N 维组合覆盖表。

算法2:

输入: F :参数集, N :维数, F_i :参数子集, $N_i(1 \leq i \leq t)$:维数

输出: $T[m, n]$:多重维数相邻因素组合覆盖表
步骤:

```

1)  $m = |NCA(m_0; N, F)|$ ;
2) for  $i = 1$  to  $t$  {  $m[i] = |NCA(m_i; N_i, F_i)|$ ;
   }
3) for  $i = 1$  to  $t$  {
4) if  $m < m[i]$  then  $m = m[i]$ ;
5) Initialize  $T[m, n]$ ;
6) if  $f_i \in F_i$  then 生成  $NCA(m_i; N_i, F_i)$ ;
7) else {生成  $N$  维  $NCA(1, 2, \dots, f_{1, N-1})$ ;
8) 生成  $NCA(m_i; N_i, F_i)$ ; } (扩展参数)
9) 扩展参数到  $f_{2, N-1}$ ; (以  $N$  维方式扩展)
10) for  $i = 2$  to  $t$  {
11) 扩展参数生成  $NCA(m_i; N_i, F_i)$ ;
12) if  $i \neq t$  then
13) 扩展参数到  $f_{i+1, N-1}$ ; (以  $N$  维方式扩展)
14) else 扩展到  $f_n$ ; } (以  $N$  维方式扩展)

```

4 结束语

提出通过精简接口数据域、确定值域的相关关系以及多重维数相邻因素组合覆盖表,层层筛选,有效减少接口测试数据的输入组,提高接口测试的效率。其中,对于精简接口数据域,可以设计软件测试工具辅助分析;对于值域的相关关系,需要测试者从功能性上着手,人工进行分析,存在一定的不确定性;作为组合测试的一种特例,对于只在相邻接口因素间存在相互作用的软件接口测试,相邻因素组合测试可以达到使用最少的测试用例得到最优测试结果的效果,但对于不具备这种特性的接口,则需要使用传统的组合测试方法进行测试。

参考文献:

- [1] 樊 玮,朱 贺. 软件结构化测试用例自动生成方法[J]. 计算机技术与发展, 2010, 20(5): 26-28.
- [2] 孙淑香,侯秀萍. 基于树型模型的黑盒测试用例自动生成[J]. 计算机技术与发展, 2009, 19(2): 77-80.
- [3] Kuhn D R, Reilly M J. An investigation of the applicability of design of experiments to software testing[C]//Proceedings of the 27th NASA/ IEEE Software Engineering Workshop. [s. l.]: NASA Goddard Space Flight Center, 2002: 91-95.
- [4] Kobayashi N, Tsuchiya T, Kikuno T. A new method for constructing pairwise covering designs for software testing[J]. Information Processing Letters, 2002, 81(2): 85-91.
- [5] Williams A W. Software component interaction testing: Coverage measurement and generation of configurations[D]. Ottawa Carleton: Institute for Computer Science, School of Information Technology and Engineering, University of Ottawa, 2002.
- [6] Cohen M B, Colbourn C J, Collofello J S, et al. Variable strength interaction testing of components[C]//Proceedings of the 27th International Computer Software and Applications Conference (COMPSAC2003). Dallas TX: [s. n.], 2003: 413-418.
- [7] Cohen M B, Colbourn C J, Ling A C H. Augmenting simulated annealing to build interaction test suites[C]//Proceedings of the 14th International Symposium on Software Reliability Engineering (ISSRE 2003). Denver Colorado: [s. n.], 2003: 394-405.
- [8] Cohen M B, Colbourn C J, Gibbons P B, et al. Constructing test suites for interaction testing[C]//Proceedings of the International Conference on Software Engineering (ICSE2003). Portland OR: [s. n.], 2003: 38-48.
- [9] Kuhn D R, Wallace D R. Software fault interaction and implication for software testing[J]. IEEE Transactions on Software Engineering, 2004, 30(6): 1-4.
- [10] Shiba T, Tsuchiya T, Kikuno T. Using artificial life techniques to generate test cases for combinatorial testing[C]//Proceedings of the 28th International Computer Software and Applications Conference (COMPSAC2004). Hong Kong: [s. n.], 2004: 72-78.
- [11] Colbourn C J, Cohen M B, Turban R C. A deterministic density algorithm for pairwise interaction coverage[C]//Proceedings of the IASTED International Conference on Software Engineering (SE2004). Innsbruck: [s. n.], 2004: 345-352.
- [12] Schroeder P J, Bolaki P, Gopu V. Comparing the fault detection effectiveness of n2way and random test suite[C]//Proceedings of the 2004 International Symposium on Empirical Software Engineering (ISESE2004). Redondo Beach, California: [s. n.], 2004: 49-59.
- [13] Colbourn C J, Martirosyan S S, Mullen G L, et al. Products of mixed covering arrays of strength two[J]. Journal of Combinatorial Designs, 2005, 14(2): 124-138.
- [14] Sherwood G B, Martirosyan S S, Colbourn C J. Covering arrays of higher strength from permutation vectors[J]. Journal of Combinatorial Designs, 2005, 14(3): 202-213.
- [15] 王子元,聂长海,徐宝文,等. 相邻因素组合测试用例集的最优生成方法[J]. 计算机学报, 2007, 30(2): 205-206.