维普资讯 http://www.cqvip.com

软件测试中的窗口测试*)

叶新铭[1] 姜树明[1] 图 雅[1,2]
(内蒙古大学计算机学院 呼和浩特 010021)[1] (内蒙古民族大学数学与计算机科学学院 通辽 028043)[2]

摘 要 当前图形界面用户窗口(GUI)在软件开发中大规模地使用,对软件测试提出了挑战。如何能够对软件的窗口进行正确的功能测试,是软件测试面临的一个重要问题。本文提出了窗口控件交互执行因果图(CICE)法,使窗口测试能够按照一定的规则进行。并且给出测试用例的生成算法,自动生成测试用例。

关键词 CICE,窗口测试,测试用例

Window Test in Software Test

YE Xin-Ming[1] JIANG Shu-Ming[1] TU Ya[1,2]
(College of Computer Science, Neimongol University, Hohhot 010021)[1]
(College of Mathematics and Computer Science, Inner Mongolia University, TongLiao 028043)[2]

Abstract Now the large-scale use of the Graphical User Interface(GUI) in software, propoesd the challenge to the software test, How Can correctly do function test alout Control's Interaction in window, this is an important issue in the software test. This paper proposed Control Interaction Cause-Effect Graph (CICE), enables the window test to defer to certain rule to carry on the test. And gave the test case production algorithm, the test case was created automatically.

Keywords CICE, Window test, Test case

1 引言

当今大部分用户软件都使用图形界面用户窗口(GUI)。一个窗口包含许多的控件,控件之间有着复杂的关系。如果没有一个合适的方法作为指导,仅仅依靠设计说明书,很难对一个窗口进行一个完整的测试[1]。采用界面构件关联图,测试用例的设计以关联图为标准,测试过程受到关联图的规范,进行以界面为基础的功能测试。但是,首先该方法的关联图构建比较复杂。第二,不能进行自动化测试用例生成[2]。在基于可视化编程环境的软件测试提出了窗口控件交互执行因果图的思想进行窗口测试,但是没有给出形式定义和算法。文[3,4]给出了软件系统测试建模和测试用例生成方法,但是不适用于GUI测试。

本文提出了根据窗口控件因果的关系,得到的窗口控件交互执行因果图(CICE),给出了测试用例的自动化生成算法,并给出相关的测试生成方法。

2 CICE图定义

在GUI面向对象环境中,窗口是软件执行的表现形式。窗口控件交互执行因果图(Control Interaction Cause Effect Graph,简称CICE)把窗口上拥有的控件分为输入、动作、输出3种类型,由此产生测试用例。

定义1 窗口CICE是一个有向无环图,它是界面执行时各控件状态由消息驱动产生变迁转移的图形描述,可用四元组 $G=(S,M,T,E)$ 表示,其中 S 是初始源状态即前置条件集,M 是消息或触发事件的有限集合,T 为目标状态集,且 $T=\{t|t\in M(S)\}$,有向边的有限集合 E 表示执行时序或约束,$E=\{e|e\in(S\times M)\cup(M\times T)\}$。

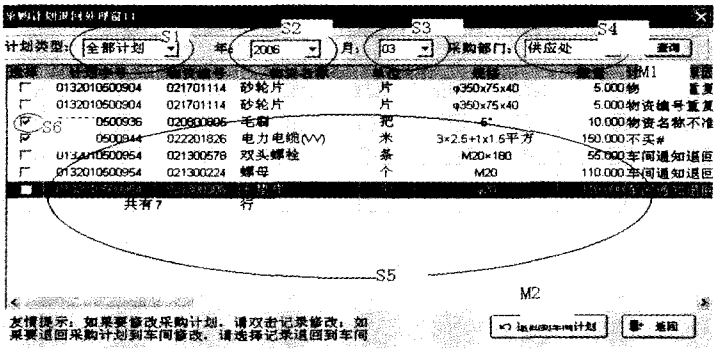

图1 一个图形窗口

图1是一个图形窗口,其中 S_1、S_2、S_3、S_4、S_6 都属于输入控件,S_5 是显示查询结果的控件属于输出控件,但是它又是 M_2 事件的输入控件,所以 S_5 作为输入控件。

在图2中,S结点上面虚框围住的为输入类,下面虚框围住的T结点为输出类;M结点表示一个动作,例如鼠标单击或按回车键等在输入类结点中也可能包含有显示结果的控件,如编辑框控件等,而同一个控件亦可能既是输入也是输出,在图中用带箭头线表示。在实际使用中,应明确结点S、T

*)国家自然科学基金(60563004),内蒙古科技攻关项目(2002061002)。**叶新铭** 教授,博士生导师,主要研究方向:计算机网络;**姜树明** 硕士研究生,主要研究方向:计算机网络;**图 雅** 硕士研究生,主要研究方向:计算机网络。

维普资讯 http://www.cqvip.com

的具体名称，M结点所对应的动作是哪个控件的哪个事件方法，为形成测试覆盖提供准确的依据。

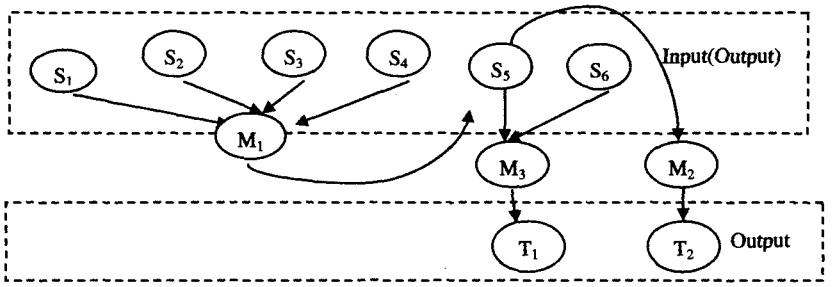

图2 CICE图

3 覆盖准则

CICE方法是一种黑盒功能性测试技术，从其所具有的组成元素我们可以获得以下几个基本测试覆盖准则。

(1)动作覆盖准则。该准则要求窗口及其控件的每一个动作(对应于CICE图中的M类结点)都至少发生一次，这是CICE测试的最基本准则。

(2)基本路径覆盖准则。所谓基本路径是指从一个输入点到一个输出点的简单路径。该准则要求CICE图中每条基本路径都至少经过一次。

(3)测试树覆盖准则。

(4)输入点与动作结点组合覆盖准则。该准则根据CICE图中所有输入点的典型取值形成一个输入值集合，按窗口测试说明书规定的动作要求，使得各个输出点对应的相关输入点及动作的所有可能组合都至少出现一次。

定义2 测试树是以CICE图中T结点为根结点，经相应的动作结点回溯至所有相关的输入点，从而形成的一个倒置的树型结构。

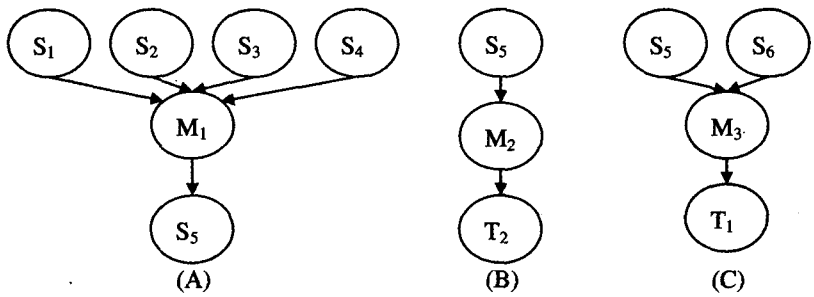

图3 测试树

图3的三棵测试树，(A)表示通过输入或选择列表框的数据组合点击查询得到查询出来的退回计划显示在数据窗口中，(B)表示双击退回计划的纪录修改退回计划，(C)表示选中退回计划的纪录，点击“退回到车间计划”，显示退回到车间计划的结果。

CICE图可重复分割成若干棵测试树(如图2所示的便为图3所对应的测试树)，测试树覆盖准则要求每一棵树都至少执行一次。可以认为，该准则是准则(1)的细化。在实际使用中，我们可以以一个动作结点为根据，将该动作产生的所有输出看成是一个结点(根)，这样获得的测试树的数目与窗口上的所有动作(事件)数一致，这将更有利于实现各个动作发生的先后顺序，以形成更加全面的测试用例。

不难看出，准则(1)和准则(2)是测试的基本要求；准则(3)是一种操作性很强的测试覆盖准则，它包含了准则(1)和准则(2)，它们都是属于路径测试技术；而准则(4)则是属于逻辑覆盖法，这是一种查错能力十分强的测试。从准则(1)～(4)的测试强度是依次递增的。在实际测试中，典型方法是将准则(3)，(4)结合起来使用或者是准则(1)，(3)，(4)结合起来使用。由于一般窗口上的控件数不会太多，所以其测试量一般是可以接受的，因而这种方法具有极高的实用性。

4 生成测试树算法

```
Input: the adjacency list of directed acyclic graphs。
Output: The array of test tree
//遍历有向图得到入度为0的结点加入到链表LFirst。
Find all vertexes which in-degree is 0 to join in list LFirst;
Create a new stack;
For (Node_1=LFirst;Node_1! =NULL;Node_1=Node_1->next)
{
  Node_2=Node_1;
  Push Node_2 into stack;
  If Node_2 has an edge (Node_2, V) and V vertex was not visited,
  push V into stack; Node_2=V;
  While (Node_2! =Node_1)
  {
    //取栈顶结点，如果栈顶结点有未被访问的兄弟结点，将兄弟结点入栈；没有，将该结点出栈
    Set Node_2 equal to the top vertex of stack;
    If Node_2 is not visited,
        If Node_2 has brother vertex Node_3 which was not visited,
        push Node_3 into stack;
        Else pop Node_2 out stack;
    //如果结点为T，则输出测试树，将栈顶结点出栈。再判断出栈的结点是否有未被访问的兄弟结点，如果有，将兄弟结点入栈。
    If Node_2's type= ="T", Output test tree, set Node_2 visited,
        and pop Node_2 out stack;
    If Node_2 has brother vertex Node_3 which was not visited, push
    Node_3 into stack;
    //如果结点为S，如果现在栈中的长度大于3，则输出测试树；如果该结点还有出弧，则把下个结点入栈，没有，则该结点出栈，再看是否存在未被访问的兄弟结点，如果有，则兄弟结点入栈。
    If Node_2's type= ="S", Set Node_2's visited times add 1;
        If Node_2's visited times equal to Node_2's in-degree, Set
        Node_2 visited;
        If length of stack>=3, Output test tree;
        If Node_2 has an edge (Node_2, V) and V vertex was not
            visited, push V into stack; Else pop Node_2 out stack,
        If Node_2 has brother vertex Node_3 was not visited, and
            push Node_3 into stack;
    //如果结点为M，为每个M结点创建保存有弧到自身结点的S结点的父亲链表。如果该结点的访问次数等于入度，将下一结点入栈，并判断自己的父亲链表是否为空，不为空，则将上一个结点加到父亲链表中；如果该结点的访问次数不等于入度，则将上一个结点加到父亲链表中，该结点出栈，如果有兄弟结点，兄弟结点
```

维普资讯 http://www.cqvip.com

```
    入栈。
    If Node_2's type= ="M", Set Node_2's visited times add 1;
        If Node_2's visited times equal to1, Create a new list for
        Node_2;
        If Node_2 's visited times equal to Node_2's in-degree,
            If Node_2 has an edge (Node_2, V) and V vertex was
                not visited, push V into stack, set Node_2 visi-
                ted;
            If Node_2's list is not NULL, Insert Node_2's father
                vertex to Node_2's list;
        Else Insert Node_2's father vertex to Node_2's list; pop
        Node_2 out stack;
            If Node_2 has brother vertex Node_3 was not visited,
                push Node_3 into stack;
    }
    Clear stack;
}
```

5 生成测试用例

CICE 测试用例分别用每一棵测试树所对应的测试用例来表示。

定义 3 CICE 测试用例表是显示测试数据及期望结果的表格。它每一行由形如 $d_b d_1 \cdots d_n d_e$ 的数据序列组成，其中 d_b 表示根结点的初始状态，$d_i(i=1,2,\cdots,n)$为一个控件的输入数据或操作，d_e 表示期望结果。

每一个形如 $d_b d_1 \cdots d_n d_e$ 的数据序列都是一个测试用例，一棵测试树可以生成多少个测试用例，要根据每一个输入包含多少典型值和多少个输入进行计算。假设有一棵测试树有 3 个输入；根据测试数据的选取方法(等价划分、边界值分析、错误推测……)，输入的典型值个数为 2、3、2，那生成的测试用例个数为(2×3×2=12)12 个。期望结果可以参照设计说明书，根据输入值的组合情况得到。下面是根据测试树生成测试例的算法。

```
Input: Get values from all the arrays of test tree and data[]; put
       these values into the array Value[N][M], and put the each
       test tree input's start place in Value[N][M]into the array
       Place[N][K];
Output: Combination[][K];
Declare int S[][K]; //save Value[N][M]'s column index
For(tree; tree<N; tree++)
{
  Set the array S[tree][]=-1;
  Compute NUM equal to combination's count;
  While (totalcnt<NUM)
  {
    If S[tree][k]==-1,set S[tree][k]=Place[tree][point];
        If S[tree][k]<K-1,k++, point++;
    Else
        If S[tree][k]= K-1, save into Combination[][K];
            totalcnt++;
            If S[tree][k]<Value[N][M]'s the last column index
                S[tree][k]++;
            If point<K-1, k++;
            Else
                Point--; S[tree][k]=-1; k--;
        Else
            If S[tree][k]<[tree][point+1]-1; S[tree][k]++;
            k++;point++;
            Else
                If k==K-1, save into Combination[][K]; total-
                cnt++;
                S[k]=-1; point--; k--;
  }
}
```

假设在 CICE 图中的输入结点包含信息输入的典型值。针对图 3(A)的测试树，测试数据选取为：S_1={ALL_SCH(所有计划)、MON(月计划)、TEM(临时计划)}，S_2={2005}，S_3={1、12}，S_4={EQU(设备处)、SUP(供应处)}。根据生成算法的到 12 组测试数据{ALL_SCH＃＃2005＃＃1＃＃SUP、ALL_SCH＃＃2005＃＃1＃＃EQU、ALL_SCH＃＃2005＃＃12＃＃SUP、ALL_SCH ＃＃2005＃＃12＃＃EQU、TEM ＃＃2005＃＃1＃＃SUP、TEM ＃＃2005＃＃1＃＃EQU、TEM ＃＃2005＃＃12＃＃SUP、TEM＃＃2005＃＃12＃＃EQU、MON＃＃2005＃＃1＃＃SUP、MON＃＃2005＃＃1＃＃EQU、MON＃＃2005＃＃12＃＃SUP、MON＃＃2005＃＃12＃＃EQU}

表 1 测试用例表

d_b-S_5(初态)	d_1-S_1(计划类型)	d_2-S_2(年)	d_3-S_3(月)	d_4-S_4(采购部门)	db- S5 (终态)
ALL_Back	ALL_SCH	2005	1	SUP	2005-1SUP 退回 ALL_SCH
ALL_Back	ALL_SCH	2005	1	EQU	2005-1EQU 退回 ALL_SCH
ALL_Back	ALL_SCH	2005	12	SUP	2005-12SUP 退回 ALL_SCH
ALL_Back	ALL_SCH	2005	12	EQU	2005-12EQU 退回 ALL_SCH
ALL_Back	TEM	2005	1	SUP	2005-1SUP 退回 TEM
ALL_Back	TEM	2005	1	EQU	2005-1EQU 退回 TEM
ALL_Back	TEM	2005	12	SUP	2005-12SUP 退回 TEM
ALL_Back	TEM	2005	12	EQU	2005-12EQU 退回 TEM
ALL_Back	MON	2005	1	SUP	2005-1SUP 退回 MON
ALL_Back	MON	2005	1	EQU	2005-1EQU 退回 MON
ALL_Back	MON	2005	12	SUP	2005-12SUP 退回 MON
ALL_Back	MON	2005	12	EQU	2005-12EQU 退回 MON

根据生成的测试数据参考设计说明书，写出了表 1 的测试用例表。当然生成测试用例表时，要注意到测试树的前后关联顺序，(A)的测试用例的生成要在(B)、(C)前，因为(B)、(C)要使用到(A)生成结果。

有了表 1 测试用例表，也就得到了测试用例中主要信息。可以根据表 1 的测试用例表填写测试用例。以测试用例表第一行记录为例，表示数据窗口起始状态为显示所有退回记录，计划类型选择全部计划，年选择 2005，月选择 1，采购部门供应处，点击查询按钮，预期结果为数据窗口显示 2005 年 1 月份供应处所有退回计划。其生成的测试用例，如表 2。

在本例的实际测试中，本窗口的 3 棵测试树共得到 16(12+1+3)个测试用例。得到测试用例后，测试人员按照测试用例手工执行测试。当然也可以使用自动化测试工具录制测试脚本，执行自动化测试。

6 应用 CICE 图法进行测试的效果分析

目前，对于大部分的带有图形用户界面的软件测试都是根据设计说明书手工编写测试用例。即使是自动化测试，其测试用例往往也是手工编写，然后根据测试用例进行脚本录制。

(1)一个系统的测试用例往往达到上万个，所以手工编写这些测试用例不但耗时，而且不同的人编写的测试用例往往格式不统一、不规范。根据 CICE 图法可以自动生成测试用例。要生成测试用例，只需要根据设计说明书和系统界面构造 CICE 图，对输入控件选择典型的测试数据，然后进行 CICE 图进行正确性检查。将通过了正确性检查的 CICE 图，使用自动生成算法可以自动生成测试用例表。根据生成的测试用例表编写测试用例，这样大大节省编写测试用例的时间。

(下转第 300 页)

维普资讯 http://www.cqvip.com

相连，块大小为32B，伪随机替换算法，2级cache 512k，4路组相连，块大小为32B，伪随机替换算法，内存256M。一级cache命中延迟为4拍，二级cache命中延迟为17拍，二级cache miss访存的延迟为30拍。编译器为ORC[12] for MIPS版本。实验基准数据的选项为O3＋SWP。测试程序为SPEC CPU 2000[13]中的整数和浮点的例子。

表1 各种调度策略的实验结果

	Base	Version1	Version2	Version3	CSS
mcf	431	436	432	434	429
bzip2	273	290	286	292	286
parser	279	278	279	278	280
swim	383	383	385	385	388
mgrid	211	210	213	211	212
applu	243	254	248	249	249
art	448	466	468	467	437
equake	339	331	341	328	333
ammp	405	406	407	407	369
Geomean	324.28	328.85	329.30	328.40	322.54

实验结果如表1所示，这些调度策略都修改依赖图的load指令的延迟，并且都修改依赖于delinquent load指令的指令的延迟。

Base是按照原有的调度策略，但不修改依赖于delinquent load指令的指令延迟。

Versonl是按照关键路径长度最优先的策略来调度的。

Version2是按照原有的调度策略来调度的。

Version3是按照长延迟优先，关键路径长度次之的调度策略来调度的。

CSS是文[10]中的调度策略。

Base和Version2相比，修改依赖于delinquent load指令的指令延迟，性能明显提高，bzip2提高4.8%，applu提高2%，art提高4.5%，平均提高1.5%。Version2性能最佳，按照关键路径长度的调度Version1和Version3次之，这是因为长延迟优先的策略更容易提升存储级并行度。CSS策略是性能最差的一个策略，主要是因为我们的平台是乱序执行的机器，原来的参数不适应乱序执行的机器。

6 相关工作

文[11]根据profile工具DCPI来收集访存指令的miss与命中信息，然后根据这些信息来进行load指令的调度。在没有性能监视工具的机器上，显然这种方法就没法用。文[10]用编译器插桩函数方式收集profile信息，但是CSS调度策略只适用于VLIW型的机器，实验表明在乱序机器上我们的方法优于CSS算法。

Bernstein[1]提出了超标量机器上的全局指令调度技术。Kerns[2]在访存延迟不确定的情况下，提出balanced scheduling来调整访存延迟，提高指令级并行度。

结论 为了提高乱序执行机器上的存储级并行度，我们引入cache敏感的调度技术，将依赖图中频繁miss的访存指令的延迟调长，并修改调度的策略，使得依赖于频繁miss指令的指令优先级最低。实验结果显示，这样调度有很好的效果，平均性能提高1.5%，个别例子的性能提高4.8%。

参考文献

1 Bernstein D. Global Instruction Scheduling for Superscalar Machines. SIGPLAN, 1991

（下转第311页）

（上接第285页）

表2 测试用例

测试编号 Plan4.1.1
测试项目 计划管理的计划退回处理窗口
测试目的 保证能够通过输入条件查询出正确的数据
测试配置 物资信息管理系统能够正常登录，计划退回窗口能够正常打开，并且各个输入和输出控件都显示了正确的数据。
测试步骤 1 检查退回计划记录数据窗口是否正确显示了所有的退回计划 2 计划类型、年、月、采购部门的下拉框分别选择全部计划、2005.1.供应处 3 点击查询按钮
预期测试结果 在退回计划记录数据窗口显示2005年1月份供应处所有的退回计划

(2)如果没有一种规则为依据进行测试用例编写，面对着窗口中众多的控件，要生成覆盖率高的测试用例集很困难。CICE图法给出了根据控件之间的因果关系生成CICE图，将众多的控件联系起来，生成了覆盖率高的CICE图。所以CICE图减少了窗口测试的随意性，并提高了测试用例覆盖率。

(3)对于测试用例的回归，如果是手工编写的测试用例，在程序修改后，需要找到所有与修改程序有关测试用例进行修改，修改的数目可能很多。如果是根据CICE图法生成的测试用例，可以修改CICE图再生成测试用例覆盖原来的测试用例即可。大大增加测试用例的可维护性，减少了修改测试用例的时间。

(4)在实际的测试中，CICE图法有很强的可用性。比FSM测试模型实际操作性强。

有时候几个关联的窗口也可以进行CICE图的构建，只要能够准确地作出CICE图，就可以自动生成测试用例。

结束语 在图形用户界面的系统中，使用CICE图法可以对窗口进行有效的功能性测试，本文给出了CICE图的形式化定义以及测试用例生成算法。依据本文的内容和算法可以设计窗口自动化测试软件，软件的CICE图的绘制方法可以按照Rational Rose的动态图的设计方法进行，且需要的变量可以保存于图形的变量结构中，这样可以使用本文的算法进行测试用例的自动生成。

参考文献

1 杜栓柱，谭建荣，陆国栋．基于界面构件关联图的软件功能测试技术．计算机研究与发展，2002，39(2)：148～152

2 金义富．基于可视化编程环境的软件测试．华南理工大学学报，2002，30(7)

3 White L. Almezen H. Generating Test Cases for GUI Responsibilities Using Complete Interaction Sequences. In: Int Symp on Software Reliability Engineering, San JoseCA. 2000-10. 110～121

4 Rosaria S, Robinson H. Applying models in your testing process. Information and Software Technology, 2000, 42(12): 815～824