基于UML的面向对象的软件测试方法

The Software Testing Methods of Object State Based on UML

何 涛[1],刘文杰[2],张学斌[3]

He Tao[1],Liu Wenjie[2],Zhang Xuebin[3]

(1.上海大学计算机学院,上海 200072;2.南京信息工程大学计算机科学与技术系,江苏南京 210044;3.上海新华控制工程有限公司,上海 200245)

(1. Coll. of Computer, Shanghai Univ., Shanghai, 200072, China; 2. Dept. of Computer Sci. & Tech., Nanjing Univ. of Info. Sci. & Tech., Nanjing, Jiangsu, 210044, China; 3. Xinhua Control Engineering Co. Ltd of Shanghai, Shanghai, 200245, China)

摘要:分析UML状态图的组成、标准事件和嵌套、并发的优点,给出利用UML状态图产生测试用例的软件测试方法。该方法可以把状态的复杂度控制在和状态属性相关的线性级别,可以尽早发现与状态相关的错误。

关键词:软件测试 面向对象 状态图 UML语言

中图法分类号:TP311.5 **文献标识码**:A **文章编号**:1002-7378(2005)04-0241-03

Abstract: An example is cited to account for an object-oriented software testing method based on UML, which constructs testing use-case according the state that the objects are being and the transfer among object states. The method can control the complexity of state within the linear scope related with state attribute, and it can find the faults and mistakes related with state earlier.

Key words: software testing, object-oriented, statechart, UML

基于对象状态的测试是面向对象软件测试中的一个重要方面,它根据被测试的类的对象所处的状态以及状态之间的转移来构造测试用例。与传统的控制流和数据流测试相比,它侧重于对象的动态行为,这种动态行为依赖于对象的状态。测试对象动态行为能检测出对象成员函数之间通过对象状态进行交互时产生的错误[1]。

以往的基于对象状态的测试方法是采用扁平状态机和状态迁移图。扁平状态机虽然能很好地提示出一些类中的错误,但不适用于面向对象的软件测试,原因是测试过程中,随着类的状态属性的增加,对象状态的数目会迅速膨胀,大大增加测试的复杂度[2]。状态转移图用于刻画对象响应各种事件时状态发生转移的情况[3],容易借助于自动机理论来选择测试时所用的事件序列和预测对象的状态变化结果(序列),但是,它难于描述继承的对象动态行为、并发的动态行为以及由数据成员和成员函数构成的对象状态和对象状态转移[4]。利用基于UML的状态图可以弥补上述缺点,可以把状态的复杂度控制在和状态属性相关的线性级别,可以尽早发现与状态相关的错误。

1 UML状态图

UML状态图中的状态是由状态名、状态变量和活动三部分组成[5]。以自动售货机类CoinBox为例,图1是它的UML状态图,图中1、2、3是售货机所处的允许卖出状态、当前状态和初始状态;$S0$、$S1$是能否出售的状态名,其冒号后是其状态变量;addQtrs()、vend()、Reset()等是投币、贩卖、重置等活动。状态变量(属性)是状态图所显示的这些类推属性,有时它还可以是临时变量,如计数器等。活动部分列出在该状态时要执行的事件和动作。在活动区中可使用3个标准事件:entry, exit和do。entry事件用于指明进入该状态时的特定动作;exit事件用于指明退出该状态时的特定动作;do事件用于指明在该状态中时执行的动作。

收稿日期:2005-03-31
修回日期:2005-09-03

作者简介:何 涛(1973-),男,广西靖西人,讲师,在读博士研究生,主要从事软件体系结构、软件形式化软件测试研究。

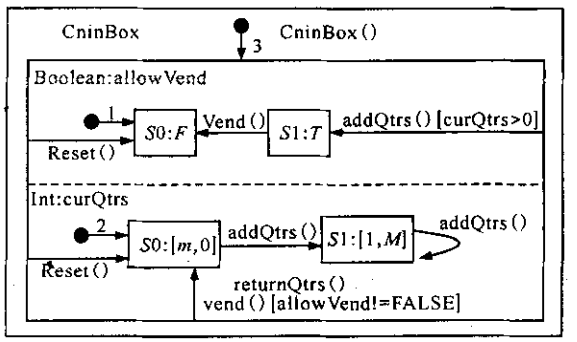

图1 类CoinBox的UML状态

UML状态图中引起状态迁移的原因通常有两种，一种是在状态图中相应的迁移上未指明事件，这表示当位于迁移箭头源头的状态中的内部动作(包括entry，exit，do以及用户定义的动作)全部执行完后，该状态迁移被自动触发；另一种是，当出现某一事件时会引起状态的迁移，在状态图中把这种引起状态迁移的事件标在该迁移的箭头上。状态迁移的形式化语法为：

event _signsture[guard _condition]/action _expression ^ send _clause

其中事件特征event _signature是由事件名后用括号括起来的参数表组成，它指出触发迁移的事件以及与该事件相连接的附加数据。警戒条件guard _condition是一个布尔表达式，如果状态迁移中既有事件又有警戒条件，则表示仅当这个事件发生并且警戒条件为真时，触发状态迁移。动作表达式action _expression是一个触发状态迁移时可执行的过程表达式，表达式中可引用该状态所拥有的对象中的属性、操作，或事件特征中的参数。发送子句send _clause是动作的一种特殊情况，用来说明在两个状态的迁移期间发送的消息。

UML状态图主要的优点在于它支持嵌套和并发。一个UML状态图中的状态可以有嵌套的子状态，一个状态所拥有的子状态可以画成另一张状态图。子状态可以是“或”关系的子状态或者是“并”关系的子状态。“或”子状态表示在任一时刻这些子状态中只有一个子状态为真；“并”子状态表示一个状态可以有多个并发的子状态，并发子状态之间用虚线分隔，用虚线分隔的每个区域表示一个并发的子状态，它有一个名字(任选)，并有一个内部的状态图。如在图1中，它把状态属性curQtrs和allowVend看成2个并发的子状态，从而可以把状态图的复杂度控制在线性级别上。

并发状态图中一个事件可能引起多个子状态的状态迁移，如图2所示。

假定图2为被测试类的X的状态图，图中有$S1$、$S2$和$S3$三个并发的子状态，其初始状态分别为

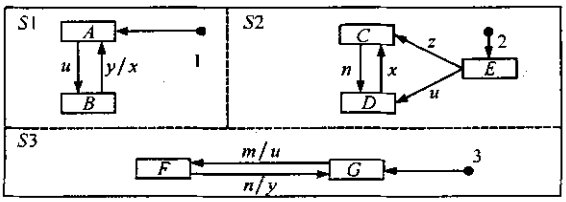

图2 并发的UML状态

A、E和G，则类X的初始状态可以用三元组(A,E,G)来表示。如果在初始状态时发生了事件m，那么在子状态$S3$中发生由状态G到状态F的迁移，同时触发事件u；由于并发的关系，事件u便在子状态$S1$和$S2$中也分别发生由状态A到状态B以及由状态E到状态D的迁移。所有，现在整个类X的状态为(B,D,F)。如果此时又产生了事件n，则子状态$S3$中发生由状态F到状态G的迁移，同时触发事件y；事件y使得子状态$S1$中发生由状态当前状态B到状态A的迁移，同时触发事件x；同理，事件x使得子状态$S2$中的状态由D迁移到C；经过一连串的触发之后，类X的状态变为(A,C,G)。

2 利用UML状态图产生测试用例

下面构造一棵测试树，根据UML状态图来自动产生测试用例。

测试树的每个节点代表对象的状态，边表示状态间的迁移，树的根节点代表对象的初始状态。首先为测试树构造一个类TestTree：

```
class TestTree {
    char[] treeNode;    //节点标识
    int    nodeLevel;   //节点在测试树中的层
                        //数
    Transition[]t;      //从本节点出发的状态迁
                        //移
    TestTree[]  childTree;  //对应状态迁移
                            t
                            //的目标状态
}
```

其中Transition是为状态迁移单独构造的类，它只是填充状态迁移的数据，在这里不再给出其具体实现。构造UML状态图测试树步骤如下：

步骤1：首先构造一个队列Queue来存放测试树的各个节点。

步骤2：定义根节点TestTree root，同时把节点标识treeNode置为对象的初始状态，nodeLevel置为0，t和childTree置NULL，把root放入队列中。

步骤3：取得队列头部的节点设为head，搜索从

head节点所对应的状态(head.treeNode)出发的状态迁移以及迁移至的目标状态,分别填充head.t和head.childTree,即把迁移至的状态作为节点head的子节点;同时置好各个子节点的属性值,其中nodeLevel为head.nodeLevel+1,从root节点开始层次遍历测试树(从第0层至head.nodeLevel层),如果在head的子节点中存在某个节点s,其所对应的状态已在第0层至head.nodeLevel层中出现过,则该节点s不再扩展,即为叶子节点。把其它没有出现过的子节点加入到队列尾部。

步骤4:head指向队列中的下一个节点,重复第二步,直至队列为空。

在步骤3中,如果某个迁移对应的目标状态已经在测试树中出现过,就不再考虑这个状态,所有并不加入到队列尾部。这样就可以有效地避免了重复构造节点,同时又不降低测试的覆盖率。通过上述步骤就可以构造出UML状态图对应的测试树。

测试树构造算法能很好地支持多个并发的子状态的情况,只是节点标识为并发子状态的合集;如果某个事件触发其它事件从而引起一系列的状态迁移时,只要把最终的状态作为节点加入到测试树中。另外,本算法同样适用于除UML状态图之外的其它状态图,如扁平状态机或D.Kung的COSD。

通过本算法构造出的CoinBox类的测试树如图3所示。图3测试树中的非黑体符号节点表示该节点代表的状态已经在之前出现过,所以不再扩展从而成为叶子节点。通过测试树可以很容易的构造出测试用例。从根节点开始沿着各个分支往下直到叶子节点,每条这样从根节点开始到某个叶子节点结束的路径上的事件按顺序组合在一起,就成为基于对象状态测试的一个测试用例。那个造成错误的事件序列很容易地被检测出来,如图3中黑体事件就是被检测出来的错误事件。

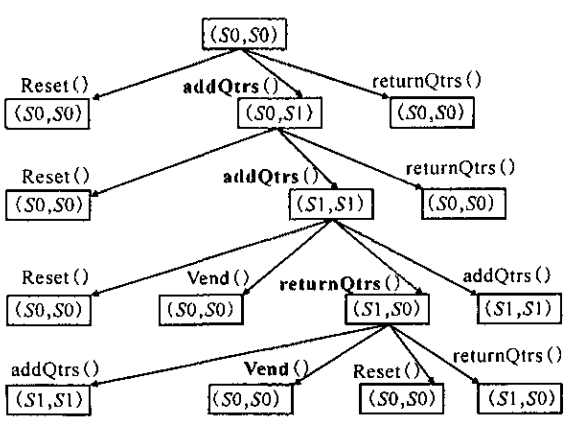

图3 类CoinBox的测试树

3 结束语

UML的状态图支持嵌套和并发,把状态的复杂度控制在和状态属性相关的线性级别;其次UML状态参数图是在面向对象软件开发的生命周期中的早期设计阶段确定的,是对对象状态的完整的描述,并不依赖于源代码,既保证了状态描述的完整性,又可以在开发早期进行测试,尽早发现与状态相关的错误,避免将错误带入到后面的开发阶段。因此可以用UML的状态图来产生有效的测试用例,这大大提高了测试的灵活性和有效性,是一种实际可行的方案。

参考文献:

[1] 夏 耘,林 华.面向对象测试技术的研究与应用[J].计算机应用与软件,2002,(2):17-20.

[2] John D. McGregor David A, Sykes. 面向对象的软件测试[M].杨文宏,李新辉,杨 洁,等译.北京:机械工业出版社,中信出版社,2003.235.

[3] Binder R V. Testingof Object-oriented System[M].北京:人民邮电出版社,2001.157.

[4] Dirk Seiferk, Steffen Helke, Thomas Santen. Comformance testing for statecharts[R]. Technical report 03-01: Technical university of berlin, 2003.

[5] OMG. Unified modeling language specification[Z]. version 1.4.2001.

(责任编辑:韦廷宗 邓大玉)

(上接第240页)

[3] 卢正鼎,陈 光.一种基于两级事务模型的先提交协议[J].华中理工大学学报,1999,27(9):101-104.

[4] 王 韬.一种新的分布事务处理模型和算法[J].军械工程学院学报,1998,10(4):51-56.

[5] Abdallah M, Guerraoui R, Pucheral P. One-Phase Commit: Does It Make Sense[A]. Proceeding of the International conference on Parallel and Distributed Systems, 1998, 182-192.

[6] 陈国宁,李陶深,廖国琼.一个带有时限的工程设计事务提交协议[J].计算机工程与应用,2004,40(14):178-180,195.

(责任编辑:邓大玉 韦廷宗)