第 33 卷 第 2 期 计 算 机 工 程 2007 年 1 月
Vol.33 No.2 Computer Engineering January 2007
·软件技术与数据库· 文章编号：1000—3428(2007)02—0044—03 文献标识码：A 中图分类号：TP311

基于状态模式的面向对象的类测试技术研究

龚红仿，童小娇
(长沙理工大学数学与计算科学学院，长沙 410076)

摘 要：类的状态测试是面向对象软件测试的重要内容，类的单元测试归结为测试类的数据成员和成员函数。该文提出了基于类的状态模式测试用例生成方法，从类的状态常量、状态变量以及成员函数等方面测试类的一致性与完整性，指出只有满足类的状态常量的约束条件与状态变量的前置条件和后置条件的测试数据，才能充分检验类的状态模式中存在的不一致、不完整性错误。

关键词：类测试；状态模式；测试用例生成

Research on Object-oriented Class Test Technology Based on State Mode

GONG Hongfang, TONG Xiaojiao
(College of Math. & Computer Science, Changsha Univ. of Science & Tech., Changsha 410076)

【Abstract】The state test of class is an important component part of object-oriented software test. Class unit test comes down to test the data member and member function. An approach of test cases generation based-on the class state model is proposed, and test consistency and integrality of classes form state constant, state variable and member function. A result is gained that only the test data which satisfies the restraint conditions of state constant and the pre-condition and post-condition of state variable fully test inconsistency and imperfect in tested classes.

【Key words】Class test; State mode; Test case generation

1 概述

面向对象的软件测试是面向对象软件开发的不可缺少的一环。面向对象程序通常由一系列类组成，在类的定义中封装了表示对象状态的数据和作用在数据上的操作。国内外许多文献都倾向于认为面向对象的软件测试应该以类为单元测试，如文献[1]讨论了面向对象软件测试的阶段划分和各阶段的测试内容，文献[2]分析了面向对象软件的特征对软件测试的影响，探讨了面向对象软件测试技术的发展状况和技术进展，文献[3]描述了类的状态测试问题，指出类测试是面向对象软件测试的关键，类的状态测试是面向对象软件测试中不可忽视的重要内容，在高质量的软件测试中起着重要的作用。但是，文献[4]认为在传统软件中单元定义的指导方针是：能够自身编译的最小程序块；单一的、独立的过程或函数；可由一个人完成的小规模工作。因而类中的成员函数可以合法地看作面向对象软件的单元，但在软件测试中又不能将类的方法测试与传统的单元测试完全等同起来，因为一个类有它自己的状态和依赖于状态的行为，类的操作既与类的状态有关，又可能改变类的状态。所以，类测试时要将其对象与其状态结合起来，测试类的状态行为，检测出类的成员函数之间通过对象的状态进行交互时产生的错误。

基于上述原因，本文将类的单元测试进一步划分为测试类的数据成员和成员函数，而不是测试类的整体，探讨将类的属性和方法分别测试的途径，提出了基于类的状态模式测试用例生成方法，从类的状态常量、状态变量以及成员函数等方面测试类的一致性与完整性，检验类的状态模式中存在的不一致、不完整性错误。

2 类的一致性与完整性测试

图 1 是用 C++伪代码描述的一个队列类模板，该队列最多可以容纳 100 个元素：

```
template <class T> class Queue
{ private:
      const unsigned max=100; unsigned elements;
  public:
    Queue <T>(){elements=0;}
     void EnQueue (const T &obj) {if (elements<max)
{Enter an obj into Queue; elements++; }}
     T & DeQueue () {if (elements>0) {Delete an obj from Queue;
elements--; }}
    bool IsEmpty(){if (elements= =0) Queue is Empty; }};
```

图 1 Queue 的一个 C++类模板

Queue 类中有一个状态常量 max，它限制队列中的最大元素个数，状态变量 elements 指队列中元素的个数，成员函数中的条件(如 elements<max)，称为状态不变式。

图 1 中的Queue类模板主要包括两个大部分：数据成员和成员函数。其中数据成员分为状态常量和状态变量；成员函数由一系列操作序列组成[5]。选择状态模式的测试用例时，一般先选定状态常量的测试用例，再选定状态变量的测试用例，最后选定成员函数所需的输入变量的测试用例。

2.1 状态常量的测试

状态常量是指在类的生命周期中必须保持的条件，为类

基金项目：国家自然科学基金项目资助(60474070)；湖南省自然科学基金项目资助项目(04JJ3031，05JJ2002)；湖南省教育厅科研基金资助项目(05C263)

作者简介：龚红仿(1968－)，男，硕士、副教授，主研方向：软件可靠性，Petri 网理论及应用；童小娇，博士、教授

收稿日期：2006-01-24 **E-mail：**gonghf@csust.edu.cn

的实例描述了一组操作界限，其定义由若干常量的类型声明及约束条件构成，可以简写成如下的谓词形式：

$$\forall C_1 : T_{C1}[P_{C1}]; \forall C_2 : T_{C2}[P_{C2}]; \cdots ; \forall C_n : T_{Cn}[P_{Cn}]$$

其中C_i(i=1,2,…,n)是状态常量的名称，T_{Ci}(i=1,2,…,n)是相应状态常量的类型，P_{Ci}(i=1,2,…,n)是关于相应状态常量的约束条件。尽管C_i(i=1,2,…,n)称为状态常量，但是在上式中，它们被看成是其值满足约束条件P_{Ci}(i=1,2,…,n)的所属类型的任意值。

定义 1（状态常量的一致性）如果存在满足约束条件P_{Ci}(i=1,2,…,n)的一组状态常量，则称该常量定义满足一致性。状态常量满足一致性时，下面的谓词为真：

$$\exists C_1 : T_{C1}[P_{C1}]; \exists C_2 : T_{C2}[P_{C2}]; \ldots ; \exists C_n : T_{Cn}[P_{Cn}]$$

定义 2（状态常量的完整性）如果常量定义的约束条件中出现的量都定义有确切的类型，则称该常量定义满足完整性。

在图 1 中只有一个状态常量max，其测试用例基于声明类型、约束条件和边界范围 3 个方面来选取满足P_C的值，得到 0、50 和 100 3 个值。该常量定义未发现不一致错误和不完整错误。

2.2 状态变量的测试

状态变量由一组类型声明及其前置条件与后置条件组成，用谓词形式表示为

$$\forall V_1 : T_{V1}[P_{V1}, Q_{V1}]; \forall V_2 : T_{V2}[P_{V2}, Q_{V2}]; \ldots; \forall V_m : T_{Vm}[P_{Vm}, Q_{Vm}]$$

也可以把状态常量、状态变量及类的不变式组合在一起形成如下谓词：

$$\forall C_1 : T_{C1}[P_{C1}]; \ldots ; \forall C_n : T_{Cn}[P_{Cn}]; \forall V_1 : T_{V1}[P_{V1}, Q_{V1}];$$
$$\ldots ; \forall V_m : T_{Vm}[P_{Vm}, Q_{Vm}] \cdot [P_C \wedge P_V, P_C \wedge Q_V]$$

其中V_i(i=1,2,…,m)是状态变量的名称，T_{Vi}(i=1,2,…,m)是相应状态变量的类型，P_{Vi}与 Q_{Vi}(i=1,2,…,m)分别表示相应状态变量的前置条件与后置条件，$P_C \wedge P_V, P_C \wedge Q_V$是类的不变式。

定义 3（状态变量的一致性）如果存在一组状态常量和状态变量的值分别满足约束条件P_{Ci}(i=1,2,…,n)和P_{Vj}、Q_{Vj}(j=1,2,…,m)，并使得类不变式 $P_C \wedge P_V, P_C \wedge Q_V$ 成立，那么该状态模式满足一致性。即下面的谓词为真：

$$\exists C_1 : T_{C1}[P_{C1}]; \ldots ; \exists C_n : T_{Cn}[P_{Cn}]; \exists V_1 : T_{V1}[P_{V1}, Q_{V1}];$$
$$\ldots ; \exists V_m : T_{Vm}[P_{Vm}, Q_{Vm}] \cdot [P_C \wedge P_V, P_C \wedge Q_V]$$

定义 4(状态变量的完整性) 如果对于已选定的满足约束条件P_{Ci}(i=1,2,…,n)的状态常量的所有组测试用例，都分别可以找到满足约束条件P_{Vj}、Q_{Vj}(j=1,2,…,m)，并使得类不变式 $P_C \wedge P_V, P_C \wedge Q_V$ 都成立的状态变量的测试用例，那么称该状态模式未发现不完整性错误。

如果对于若干组满足约束条件P_{Ci}(i=1,2,…,n)的状态常量的测试用例，则可以找到满足约束条件P_{Vj}、Q_{Vj} (j=1,2,…,m)，并使得类不变式 $P_C \wedge P_V, P_C \wedge Q_V$ 成立的状态变量的测试用例，而对于另外若干组满足约束条件P_{Ci}(i=1,2,…,n)的状态常量的测试用例，找不到满足约束条件P_{Vj}、Q_{Vj} (j=1,2,…,m)使得类不变式 $P_C \wedge P_V, P_C \wedge Q_V$ 不成立的状态变量的测试用例，则该状态模式是不完整的。当状态模式不完整时下面的谓词

$$(\exists C_1 : T_{C1}[P_{C1}]; \ldots; \exists C_n : T_{Cn}[P_{Cn}]; \exists V_1 : T_{V1}[P_{V1}, Q_{V1}]; \ldots; \exists V_m : T_{Vm}[P_{Vm}, Q_{Vm}]$$
$$\cdot [P_C \wedge P_V, P_C \wedge Q_V]) \wedge (\exists C_1 : T_{C1}[\neg P_{C1}]; \ldots; \exists C_n : T_{Cn}[\neg P_{Cn}]; \exists V_1 : T_{V1}[\neg P_{V1}, \neg Q_{V1}];$$
$$\ldots; \exists V_m : T_{Vm}[\neg P_{Vm}, \neg Q_{Vm}] \cdot [\neg P_C \vee \neg P_V, \neg P_C \vee \neg Q_V])$$

为真：

我们根据选定的 max 的测试用例分别选定状态变量 elements 的测试用例：

(1)当 max=0 时，代入不变式 $P_C \wedge Q_V$，得(0⩾0)∧(0⩽100)∧(elements⩽0)，此时 P_V 不存在。可以选定满足条件 elements⩽0 的状态变量的值，令 Queue 为空队列。

(2)当 max=50 时，代入不变式 $P_C \wedge P_V, P_C \wedge Q_V$，得：(50⩾0)∧(50⩽100)∧(elements⩾0)，(50⩾0)∧(50⩽100)∧(elements⩽50)。可以选定满足条件 elements>=0 和 elements<=50 的状态变量的值，有 3 种情况：队列空(elements=0)、队列满(elements=50)以及队列中元素个数满足 elements>0 和 elements<50 的 elements 的值，如取 elements=25。

(3)当 max=100 时，代入不变式 $P_C \wedge P_V, P_C \wedge Q_V$，得(100⩾0)∧(100⩽100)∧(elements⩽100)，此时 $P_C \wedge P_V$和$P_C \wedge Q_V$相同。可以选定满足条件 elements⩾0 和 elements⩽100 的状态变量的值，有 3 种情况：队列空(elements=0)、队列满(elements=100)以及队列中元素个数满足 elements>0 和 elements<100 的 elements 的值，如取 elements=25。

由于对于所有组已选定的状态常量的测试用例，都可以找到满足类不变式 $P_C \wedge P_V, P_C \wedge Q_V$ 的状态变量 elements 的测试用例，因此该状态模式定义未发现不一致性错误和不完整性错误。

2.3 成员函数的测试

面向对象的系统是由成员函数和状态变量相互作用而运行的[6]。用O($c_1,c_2,\ldots,c_w$; $v_1, v_2,\ldots, v_m$;$M_1,M_2,\ldots,M_k$)表示对象，其中c_i为状态常量，v_i为状态变量，M_i为成员函数，S_i($c_1,c_2,\ldots,c_n$; $v_1, v_2,\ldots, v_m$)表示对象状态，T_i(S_i,M_i，S_{i+1})表示对象O在事件 E_i的作用下由状态S_i转换到另一状态S_{i+1}的一种状态转换，其中S_i为当前状态，M_i为E_i触发的成员函数，S_{i+1}为转换后的状态，i∈[0,n])，则面向对象系统的运行方式可以表示为 $S_0 \xrightarrow{E_1/T_1} S_1 \xrightarrow{E_2/T_2} S_2 \cdots \xrightarrow{E_n/T_n} S_{n+1}$，即系统在$E_i/T_i$的作用下，实现对象状态转移。

对象的行为很大程度上依赖于对象的当前状态，即满足前置条件的状态变量的当前值。假设系统中存在下面的状态转换：

$$S_0 \xrightarrow{E_0/T_0} S_1 \cdots \xrightarrow{E_{i-1}/T_{i-1}} S_i \xrightarrow{E_i/T_i} \cdots \xrightarrow{E_{j-1}/T_{j-1}} S_j \xrightarrow{E_j/T_j} \cdots \xrightarrow{E_n/T_n} S_{n+1}$$

那么对于其中一个状态转换T_j(S_j,M_j,S_{j+1})中，M_j的执行情况受到对象当前状态S_j的某个满足约束条件的状态常量c_j和满足前置条件的状态变量 v_j的值约束，而 v_j在某一状态转移T_i中的成员函数M_i定义或者改变。这样，M_j的执行情况就依赖于M_i对 v_j作用的结果。利用这种成员函数和状态变量相互作用的关系，可以检测成员函数在这种交互作用下产生的错误。

定义 5（当前值与后值）对于某一对象 O，在 E_i/T_i 的作用下：

(1)处于S_i状态且满足约束条件P_{Ck}(k=1,2,…,n)的状态常量的值称为状态常量当前值；

(2)处于S_i状态且满足前置条件P_{Vj} (j=1,2,…,m)的状态变量的值称为状态变量当前值；

(3)处于S_{i+1}状态且满足约束条件P_{Ck}(k=1,2,…,n)的状态常量的值称为状态常量后值；

(4)处于S_{i+1}状态且满足后置条件Q_{Vj} (j=1,2,…,m)的状态变量的值称为状态变量后值。

定义 6 （成员函数的一致性）对于某一对象O处于当前

状态S_i，如果对于任意选定的一组状态常量和状态变量当前值使得类不变式 $P_C \wedge P_V$ 成立，那么在 E_i/T_i 的作用下，能够得到相应状态变量的后值且满足类不变式 $P_C \wedge Q_V$，则称该成员函数 M_i 满足一致性。即下面的蕴含表达式为真：

$$\forall C_1 : T_{C1}[P_{C1}]; \ldots; \forall C_n : T_{Cn}[P_{Cn}]; \forall V_1 : T_{V1}[P_{V1}]; \ldots; \forall V_m : T_{Vm}[P_{Vm}] \cdot [P_C \wedge P_V]$$
$$\xrightarrow{E/T} C_1 : T_{C1}[P_{C1}]; \ldots; C_n : T_{Cn}[P_{Cn}]; V'_1 : T_{V1}[Q_{V1}]; \ldots; V'_m : T_{Vm}[Q_{Vm}] \cdot [P_C \wedge Q_V]$$

其中，$\xrightarrow{E_i/T_i}$ 表示在 E_i/T_i 作用下的蕴含，V'_i表示相应状态变量V_i的后值。

反之，如果对于某一满足类不变式的状态常量当前值与状态变量当前值，在 E_i/T_i 的作用下，得到相应状态变量的后值，但不满足类不变式 $P_C \wedge Q_V$，则称该成员函数 M_i 不满足一致性。

定义 7 (成员函数的完整性) 对于某一对象O处于当前状态S_i，如果对于所有组状态常量当前值和状态变量当前值使得类不变式 $P_C \wedge P_V$ 成立，那么在 E_i/T_i 的作用下，都分别能够得到相应状态变量的后值且满足类不变式 $P_C \wedge Q_V$，则称该成员函数 M_i 未发现不完整性错误。

反之，如果对于若干组状态常量当前值和状态变量后值使得类不变式 $P_C \wedge P_V$ 成立，在 E_i/T_i 的作用下，能够得到相应状态变量的后值且满足类不变式 $P_C \wedge Q_V$，而对于另外若干组状态常量当前值和状态变量后值使得类不变式 $P_C \wedge P_V$ 成立，但在 E_i/T_i 的作用下，找不到相应状态变量的后值使得类不变式 $P_C \wedge Q_V$ 成立，则称该成员函数 M_i 不满足完整性。即下面的蕴含表达式为真：

$$\exists C_1 : T_{C1}[P_{C1}]; \ldots; \exists C_n : T_{Cn}[P_{Cn}]; \exists V_1 : T_{V1}[P_{V1}]; \ldots; \exists V_m : T_{Vm}[P_{Vm}] \cdot [P_C \wedge P_V]$$
$$\xrightarrow{E/T} \forall C'_1 : T_{C1}[P_{C1}]; \ldots; \forall C'_n : T_{Cn}[P_{Cn}]; \forall V'_1 : T_{V1}[Q_{V1}];$$
$$\ldots; \forall V'_m : T_{Vm}[Q_{Vm}] \cdot [\neg P_C \vee \neg Q_V]$$

这里，C'_i表示相应状态常量C_i的后值，V'_i表示相应状态变量V_i的后值。

表 1 是对模板类 Queue 的成员函数 EnQueue (const T &obj)的一致性与完整性测试结果。表中“－”表示不存在或不确定。从表中可以看出，对于选定的类数据成员，在成员函数 EnQueue (const T &obj)的作用下，仍满足类不变式，未发现成员函数 EnQueue (const T &obj)存在不一致和不完整性错误。类似地也可以测试其它成员函数的一致性与完整性。

表 1 成员函数 EnQueue (const T &obj)的一致性与完整性测试

状态常量 max	状态变量当前值 elements	队列元素	输入参数	满足类不变式 $P_C \wedge P_V$	状态变量后值 elements	新队列元素	满足类不变式 $P_C \wedge Q_V$
0	0	空队列	b	TRUE	1	b	TRUE
50	0	空队列	b	TRUE	1	b	TRUE
	5	$e_1,e_2,...,e_5$	b	TRUE	6	$e_1,e_2,\ldots,e_5$,b	TRUE
	50	$e_1,e_2,...,e_{50}$	b	FALSE	–	–	TRUE
100	0	空队列	b	TRUE	1	b	TRUE
	5	$e_1,e_2,...,e_5$	b	TRUE	6	$e_1,e_2,\ldots,e_5$,b	TRUE
	100	$e_1,e_2,\ldots,e_{100}$	b	FALSE	–	–	TRUE

3 结束语

本文提出了基于类的状态模式测试用例生成方法，得到只有满足类的状态常量的约束条件与状态变量的前置条件和后置条件的测试数据，才能充分检验出类的状态模式中存在的不一致、不完整性错误的结论。从文中所提出的测试方法可以看出，未发现错误的状态模式测试并不能保证状态模式的一致性与完整性，但是发现错误的状态模式测试一定能够找出状态模式的不一致和不完整之处，达到了软件测试的目的。基于状态模式的类测试尚有许多需要解决的问题，本方法往往在实际应用中较为复杂，有时还难以奏效，状态模型的分层和并发状态等机制也许可以解决这一问题，这一点有待进一步的研究。

参考文献

1 Kung D, Gao J, Hsia P. Developing an Object-oriented Software Testing and Maintenance Environment[J]. Commun. of ACM, 1995, 38 (10): 75-87.

2 金凌紫. 面向对象软件测试技术进展[J]. 计算机研究与发展, 1998, 35(1): 6-13.

3 张雪萍, 庄　雷, 范艳峰. 基于状态的类测试技术研究[J]. 小型微型计算机系统, 2002, 23(9): 1121-1124.

4 Jorgensen P C. 软件测试[M]. 韩　柯, 杜旭涛, 译. 北京: 机械工业出版社, 2003.

5 李　刚, 朱关铭. 结构化面向对象形式规格说明语言 OOZS——规格说明测试[J]. 上海大学学报, 1998, 4(4): 429-433.

6 李庆华, 刘金根, 缪天鹏, 等. 一种基于类数据流的软件测试技术[J]. 华中科技大学学报, 2003, 31(11): 17-19.

(上接第 30 页)

3 Al-Ali R, Rana O, Laszewski G, et al. A Model for Quality of Service Provision in Service Oriented Architectures[J]. International Journal of Grid and Utility Computing, 2004.

4 Ali A S, Rana O, Al-Ali R, et al. UDDIe: An Extended Registry for Web Services[C]. Proceedings of Workshop on Service-oriented Computing: Models, Architectures and Applications. Olando FL, USA: IEEE CS Press, 2003.

5 克里斯琴·蒙特. 博弈论与经济学[M]. 北京: 经济管理出版社, 2004.