面向对象软件度量C&K方法的研究与改进

马志新，徐德启，杜伟杰

(兰州大学信息科学与工程学院 兰州 730000)

【摘要】在现有面向对象软件度量方法和度量准则的基础上，结合面向对象技术特性，对C&K面向对象度量方法进行了分析和改进，提出了类的复杂性、类的方法个数等类规模的度量指标，以及多继承数深度、继承方法比、继承属性比等类的继承性度量指标，并对这些指标在工程实践中的应用进行了讨论。改进后的度量指标可以有效地补充C&K方法，优化度量结果。

关 键 词 软件度量; 面向对象; 规模; 继承
中图分类号 TP311 文献标识码 A

An Improvement on C&K Object-Oriented Software Metrics Suite

MA Zhi-xin，XU De-qi，DU Wei-jie

(School of Information Science & Engineering Lanzhou University, Lanzhou 730000)

Abstract Based on the study of object-oriented software measurement and the characteristics of object-oriented technology, some improvements on C&K metrics suite are discussed in this paper. Some new metrics for class size and class hierarchy measurement are presented including complexity per class, number of methods, depth of multi-inheritance tree and number of father etc. Meanwhile, the applications of these improved metrics in software engineering are discussed. All the improvements can supplement C&K metrics and optimize the result of measurement.

Key words software measurement; object-oriented; size ; inheritance

随着面向对象技术的广泛应用，面向对象软件度量的理论研究和实践应用已成为软件工程领域的研究热点之一[1-3]，先后出现了一批有效的面向对象的软件度量方法。本文在分析现有面向对象度量方法和度量准则的基础上，结合面向对象技术的特性，针对C&K面向对象度量方法中类规模的度量、类的继承性度量等指标和方法做出改进和完善，提出了类的复杂性、类的方法个数等类规模的度量指标，以及多继承数深度、继承方法比、继承属性比等类的继承性度量指标。并对改进后的指标和方法在工程实践中的有效应用进行了讨论。

1 面向对象软件度量方法

目前主要对象的软件度量方法有：C&K方法、MOOD方法以及Chen&Liu方法等[4-6]。MOOD方法包含4类度量算法集：(1) 封装性度量；(2) 继承性度量；(3) 耦合性度量；(4) 多态性度量。该方法由于受环境影响很大，所以应用相对不够广泛。Chen&Liu度量方法从8个方面给出了度量方法：(1) 操作复杂性度量；(2) 操作参数复杂性度量；(3) 属性复杂性度量；(4) 操作耦合度量；(5) 类的继承性度量；(6) 内聚度量；(7) 类耦合度量；(8) 重用度量，相应的度量指标考虑了面向对象技术的特点，但没有给出形式化的定义，缺乏可操作性。C&K方法给出了6个类级别的度量指标：(1) 每个类的加权方法数；(2) 继承树的深度；(3) 类的孩子数目；(4) 对象类之间的耦合；(5) 一个类的相应集合；(6) 类内聚缺乏度。该方法很好地反映了面向对象技术的特点，设计者可以根据这6个指标的度量值指导类的设计，调节设计的灵活度，维护难度之间的关

收稿日期：2003 – 12 – 02
基金项目：甘肃省自然科学基金资助项目(ZS011-A25-017-G)
作者简介：马志新(1973 –)，男，博士，副教授，主要从事软件工程与数据挖掘方面的研究.

系。但方法也存在一些局限和不足。

2　对C&K度量方法中类规模度量的分析与改进

在C&K方法中类规模的度量主要采用类的加权方法数(Weighted Methods per Class，WMC)。

定义　设类C有n个方法M_1，M_2，…，M_n，每个方法的复杂度分别为C_1，C_2，…，C_n，则$W_{\mathrm{WMC}}=\sum_{i=1}^{n}C_i$。

这里每个方法的复杂度使用传统的度量方法计算。W_{WMC}揭示了类的开发和维护花费的时间和精力。类的W_{WMC}越大，其对子类可能的影响越大，其通用性和可复用性就越差。用W_{WMC}度量一个类的规模复杂性，存在如下不足之处：(1) 没有考虑属性成员对一个类的规模复杂度的影响；(2) 没有考虑成员数目影响个体复杂性时的非线性性质；(3) 没有考虑具有最大复杂度的成员更容易影响类的复杂度的现象。为解决上述问题，在W_{WMC}定义的基础上提出如下度量指标并分别定义：(1) 类的复杂性(Complexity Per Class，CPC)：

定义　$C_{\mathrm{CPC}}=aW_{\mathrm{WAC}}+bW_{\mathrm{WMC}}=a\dfrac{N_{\mathrm{NOA}}}{N_{\mathrm{NOT}}}+b\sum_{i=1}^{n}C_i$。

式中　b为调节系数，由于一个属性和一个方法对一个类的规模复杂性的影响是不一样的，调节因子a和b可以适当的反映此差异，其值是一个通过大量的类复杂性度量实践获取的经验值，通过观察，a和b基本上保持1:4的关系，在实际度量中，b取1时，a取0.25可得到比较合理的类复杂性度量值；W_{WMC}同C&K方法中类的加权方法数的定义(Weight Attributes per Class，WAC)是类的加权属性数，N_{NOA}为类中属性(数据)成员的个数，N_{NOT}为属性成员的数据类型的个数。(2) 类的方法个数(Number of Methods，NOM):

定义　设M_i是一个度量类中的方法，则$N_{\mathrm{NOM}}=\sum M_i$。

此处N_{NOM}既包括继承来的方法个数也包括类中声明的方法个数。在度量类规模时，一个类的C_{CPC}和N_{NOM}的值大表明潜在的增加了该类的特殊性，重用受到限制；当一个类的规模复杂性过大时，表明对该类设计、编码、测试、维护的工作量相对增大，应该重点管理；同时，有可能表明该类可能抽象不当，需要分解。实践研究表明N_{NOM}大的类，多是独立的、不重用的，大多数类的方法个数都较少，基本在10个左右。

使用C_{CPC}和N_{NOM}度量一个类的规模复杂度具有如下优点：(1) C_{CPC}综合考虑了一个类的属性和方法对该类规模复杂性的影响，体现了一个类的整体封装性，指标涵盖的范围较全面；(2) 调节因子a和b可反映属性和方法对类的规模复杂性的影响差异；(3) 单独把一个类的方法个数N_{NOM}作为一个指标可以增加度量的直观性和细致性，它可反映C_{CPC}(和W_{WMC})不能直接反映的方面。例如在设计一个类时一般要求方法不能太多，如果过多则可能需要分解；而C_{CPC}(或W_{WMC})值很大时，方法个数可能很少，并不能就此说明该度量类需要分解。

3　对C&K度量方法中类的继承性度量的分析与改进

C&K方法中类继承性度量指标主要采用：继承树的深度(Depth of Inheritance Tree，DIT)和类的孩子数目(Number of Children，NOC)。

定义　DIT指对象所属类在继承树中的深度(层次)，其中根节点为0。

定义　NOC是继承树中一个类的直接孩子数目。

类的DIT大表示它可能继承的方法数目大，复用度高，但是预测它的行为将困难，同时设计复杂性大。N_{NOC}越大表示复用越好；但是当N_{NOC}增大时，父类表示的抽象可能在减弱，可能有些子类实际上不是父类的合适成员，同时测试工作量将增大。C&K方法使用DIT和N_{NOC}对类的继承性进行度量时，存在如下不足之处：(1) 无法度量多重继承，在遇到多重继承时会出现歧义。如图1所示，A为继承树中的根节点，类E多重继承了B、C和D。根据C&K方法中DIT的定义，计算类E的D_{DIT}值有两个：2和3，而且都是正确的，这就引起了歧义，在度量中是不允许的；(2) 度量一个类所能直接影响的作用域，无法度量它所能影响的整个作用域；(3) 继承包含了方法和属性的继承，虽然类封装了属性和方法，但在度量时不能用类代替属性和方法，它们的级别不同，所反映的问题也不同。例如若一个类的方法全部是继承

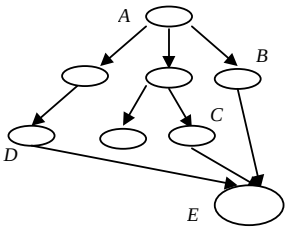

图1　一个出现度量歧义的多重继承类图

来的则说明该类重用性大，而这是用D_{DIT}和N_{NOC}反映不出来的。

综合考虑类继承的性质，在已有度量理论和方法的基础上给出继承性度量指标如下：1) (Depth of Multi-Inheritance Tree and Number of Fathers，MDIT.NOF)．(1) 设P_{PL}为类继承树中从根节点到所度量类的所有可能路径的长度，M_{MAX}为取最大值函数，则该类在继承树中的深度为：$M_{MIDT}=M_{MAX}(P_{PL})$，其中，规定树根结点的$M_{MDIT}=0$；(2) 设一个类的父类个数为n，则 $N_{NOF}=n-1$。$N_{NOF}=0$表示单重继承；$N_{NOF}=-1$则表示该结点为根结点，没有父类。考虑到在实际的设计中很少用到多重继承，单独把N_{NOF}分离出来会使度量指标显得很凌乱而且度量结果值大多相同而不具有可比性，所以此处把M_{MDIT}和N_{NOF}合为一个指标$M_{MDIT}.N_{NOF}$，这是一个实数形式，但意义不完全等同于实数。例对图1中的类E，其$M_{MDIT}.N_{NOF}=3.2$，表示类E的继承树深度为3，父类个数为3，为多重继承。在实际应用中，面向对象编程语言一般都有自己的类库，类库中的类代码对编程者来说是不可见的，所以规定对类库中的类，其$M_{MIDT}.N_{NOF}=0.0$。2) 类的所有子孙数目(Number Of All Children，NOAC)。设C_i是类C的子孙，则 $N_{NOAC}=\sum C_i$ ；3) 类的直接孩子数目，设S_i是类C的直接孩子，则 $N_{NOC}=\sum S_i$ ；4) 一个类的继承方法比(Inheritance Methods Per Class，IMPC)，设类C中继承来的方法数目为(Number of Inherited Methods，NOIM)，所有可用的方法数目为(Number of All Methods，NOAM)则 $I_{IMPC}=\dfrac{N_{NOIM}}{N_{NOAM}}$ ；5) 一个类的继承属性比(Inheritance Attributes Per Class，IAPC)，设类C中继承来的属性数目为(Number of Inherited Attributes，NOIA)，所有可用的属性数目为 (Number of All Attributes，NOAA)，则 $I_{IAPC}=\dfrac{N_{NOIA}}{N_{NOAA}}$。

上述指标涵盖了C&K方法中的D_{DIT}和N_{NOC}，并充分考虑了实际情况，可更全面度量类的继承性。其中，I_{IMPC}和I_{IAPC}大表示重用性好，但测试难度大，需要重点测试；$M_{MDIT}.N_{NOF}$反映了能对该度量类产生影响的作用域，其值大表示该类维护强度大，其超类有较大的重用性，对象间有较大的继承性耦合；$N_{NOF}=0$是设计提倡的，多重继承增加设计的灵活性，但会增加系统的复杂性；N_{NOC}和N_{NOAC}反映了该类能影响到的作用域，其值大则表示该类具有较大的重用性或该类可能抽象不当而应该重新设计，需要重点测试。采用上述五种指标，在度量类的继承性时有如下优点：解决了多重继承的度量问题，消除了歧义；考虑了面向对象语言类库的透明性对度量的影响；对一个类继承来的方法和属性的度量有助于更确切的把握该类的性质和质量，有利于对类做出准确评估；全面考虑了一个类受影响和它能够影响到的全部作用域，对继承性的度量更加全面。

4 结 束 语

本文在分析现有面向对象度量方法和度量准则的基础上，对C&K面向对象度量方法中类规模的度量、类的继承性度量等方法做出改进和完善，提出了CPC、NOM、MDIT.NOF、NOAC、NOC、IMPC、IAPC等度量指标，改进后的度量指标可以有效地补充C&K方法，优化度量结果。下一步将对C&K面向对象度量方法中耦合性度量、类的内聚性度量、多态性度量等方面进行研究，并期望提出相应的改进和完善。同时结合本文提出的改进指标和方法，进行面向对象软件度量自动化支持工具的研究[7]。

参 考 文 献

[1] Weyuker E J. Evaluating software complexity measures[J]. IEEE Transactions on Software Engineering, 1988, 14(9): 1 357-1 365.

[2] Horst Z, Thomas F. Properties of object-oriented software measures [C]//Proc. of the 7th Annual Oregon Workshop on Software Metrics, Dregon,USA: 1995: 321-330.

[3] Bansiya J, Davis C G. A hierachical model for object-oriented design quality assessment[J]. IEEE Transaction on Software Engineering, Austin Texas: 2002, 28(1):4-17.

[4] Chidamber S R, Kemerer C F. A metrics suite for object oriented design[J]. IEEE Transactions on Software Engineering, 1994, 20(6): 476-493.

[5] Abreu F B, Gonlao M, Esteves R. Toward the design quality evaluation in object-oriented software systems[C]//Proc. of the 5th International Conference on Software Quality, 1995: 44-57.

[6] Chen J Y, Lu J F. New metric for object-oriented design[J]. Information and Software Technology. 1993, 35(4): 232-240.

[7] 梅 琳, 杜晓晨, 李 茵, 等. 面向对象软件度量自动化参考模型MOOP[J]. 计算机工程与科学, 2001, (5): 39-42.

编 辑 孙晓丹