基于 PROBE 方法的 DWFC 内聚度量改进

肖 瑶[1,2] 张为群[1,2]

(西南大学计算机与信息科学学院 重庆 400715)[1] (重庆市智能软件与软件工程重点实验室 重庆 400715)[2]

摘 要 利用有向带权伪图(Directed Weighted False Chart,DWFC)表示面向对象程序中类内部成员间的依赖关系,提出一种基于 DWFC 的面向对象类内聚度量方法,结合 PSP(Personal Software Process)技术中 PROBE(PROxy-Based Estimating)规模估算方法,改进了良好内聚度量方法的验证准则,并结合实验验证了该方法的优越性。

关键词 DWFC,内聚度量,PROBE,度量验证

An Improvement to DWFC Cohesion Measurement Based on PROBE Method

XIAO Yao[1,2] ZHANG Wei-Qun[1,2]

(College of Computer and Information Science,Southwest China University,Chongqing 400715)[1]

(Chongqing Intelligent Software and Software Engineering Laboratory,Chongqing 400715)[2]

Abstract Using DWFC(Directed Weighted False Chart) to represent the dependent relationship between internal members of the classes in object-oriented programs, promoting an object-oriented cohesion measurement method based on it, combing PROBE method in PSP technology to improve the present cohesion measurement validation rules, proving the advantages of the method through experiment.

Keywords DWFC, Cohesion measurement, PORBE, Measurement validation

1 引言

内聚性是一种软件内部属性,标志模块内部各个元素彼此结合的紧密程度,通常被认为与可维护性、重用性和可靠性等外部属性具有相关性,因此高内聚成为软件开发追求的目标之一。内聚是一个较为模糊的概念,直接评价一个模块是否内聚比较困难,所以现有的方法都是从量化的角度来评价可能对内聚造成影响的一些参数从而达到间接度量内聚特性的目的。总结现有的类内聚度量方法,普遍存在的问题包括:对特殊方法的考虑、类成员之间的不同联结程度对类内聚影响程度不同、是否满足理论验证以及通过量化方法来近似刻画模块内聚的有效性等问题。

所以本文将从以下几个方面进行论述:首先介绍基于 DWFC 的面向对象内聚度量方法,理论验证该方法满足现有的度量验证准则,引入 PSP 技术中的 PROBE 规模度量方法,结合该方法改进内聚度量的验证准则,通过实验验证方法的优越性。

2 一种基于图的面向对象内聚度量方法

目前很多度量准则都依赖于具体实现语言,用图的方法可以建立独立于语言的度量模型。因此,忽略具体细节,我们首先将面向对象系统表示成如下的一个简单有向图,然后选取其中类内部方法以及属性间的相互关系来进行内聚度量。

K. Ottenstein 和 L. Ottenstein 于 1984 年首次提出软件内部关系依赖图可以用于软件工程诸多应用,因此本文采用类内部成员间的依赖关系来度量内聚特性,认为成员间的依赖关系越强,彼此间联结强度越紧密,内聚度越高。所以首先将类内部成员之间的依赖关系定义为属性与属性间,属性与方法间以及方法与方法间三种,采用有向带权伪图(DWFC)来表示类内部成员之间的依赖关系(DWFC 中仅仅包含通用方法,即对内聚不构成影响的特殊方法不参与内聚度量,M 表示方法结点,A 表示属性结点),得到如下图 2。

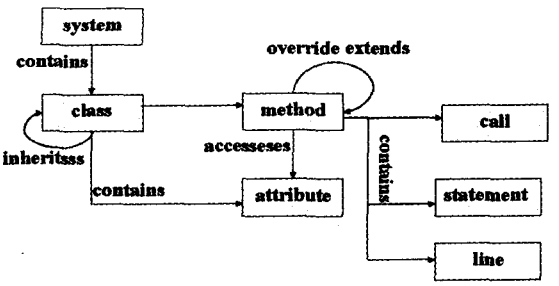

图 1 一个独立于语言的面向对象系统元模型

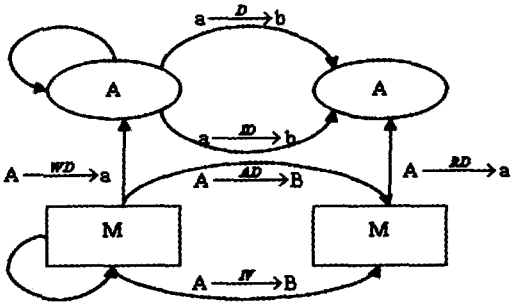

图 2 类内聚度量模型(DWFC)

2.1 对 DWFC 的定义

G〈N,E〉:用 DWFC 来表示一个面向对象系统中的类,N 表示结点集合,E 表示边的集合。

肖 瑶 硕士,主要研究领域为面向对象技术,软件度量,软件建模;张为群 教授,主要研究领域为形式化软件工程,软件测试。

η_i：表示结点类型，$\eta_i \in$ {method、attribute}。

ε_i：表示边类型，$\varepsilon_i \in \{a \xrightarrow{D} b, a \xrightarrow{ID} b, A \xrightarrow{IV} B, A \xrightarrow{AD} B, A \xrightarrow{RD} a, A \xrightarrow{WD} a\}$。

#：表示边或者结点个数。

N(type(n))：type(n)表示结点 n 的类型

E(source(e), target(e), type(e), weight(e))：source(e)表示边 e 的起始结点，target(e)表示边 e 的目标结点，tpye(e)表示边 e 的类型，Weight(e)表示边 e 的权值。

(1)对结点数目进行定义(NumofNode)

若 $n \in N$ 且 $type(n)=\eta_i$，

则 $NC(n,\eta_i,\eta_2,\varepsilon_1):=\#\{m \in N \mid type(m)=\eta_2 \cap \exists e \in E: source(e)=n, target(e)=m, type(e)=\varepsilon_1\}$；

(2)对边数目进行定义(EdgeCount)

若 $n \in N$ 且 $type(n)=\eta_i$，

则 $EC(n,\eta_i,\varepsilon_1):=\#\{e \in E\ source(e)=n, type(e)=\varepsilon_1\}$；

2.2 基于 DWFC 的依赖关系定义

定义 1(属性间的依赖关系) 对于类的成员变量，若属性 a 值的定义依赖于属性 b，则称属性 a 依赖于属性 b，记为 $a \longrightarrow b$；其中若属性 a 值由属性 b 定义，则称属性 a 直接依赖于属性 b($a \xrightarrow{D} b$)；若属性 a 的值能否被定义由属性 b 决定(一般出现在控制条件中)，则称属性 a 间接依赖于属性 b($a \xrightarrow{ID} b$)。

性质 1 属性间的依赖关系存在传递性：$a \xrightarrow{D} b, b \xrightarrow{D} c \Rightarrow a \xrightarrow{D} c$；$a \xrightarrow{ID} b, b \xrightarrow{ID} c \Rightarrow a \xrightarrow{ID} c$。

定义 2(方法间的依赖关系) 对于类的成员方法，依赖关系存在调用依赖以及属性依赖两种。若方法 A 调用了方法 B，则称 A 调用依赖于 B，记为 $A \xrightarrow{IV} B$；若方法 A 中引用的属性 a 由方法 B 定义，则称方法 A 属性依赖于方法 B，记为 $A \xrightarrow{AD} B$。

性质 2 方法间的依赖关系存在传递性：

$A \xrightarrow{IV} B, B \xrightarrow{IV} C \Rightarrow A \xrightarrow{IV} C$；$A \xrightarrow{AD} B, B \xrightarrow{AD} C \Rightarrow A \xrightarrow{AD} C$。

定义 3(方法与属性间的依赖关系) 类中成员方法以及成员变量之间存在的依赖关系主要体现在存取操作，若方法 A 仅读取了属性 a，则称 A 读依赖于 a，记为 $A \xrightarrow{RD} a$；若方法 A 修改(定义)了属性 a，则称 A 写依赖于 a，记为 $A \xrightarrow{WD} a$。

定义 4(零 DWFC 图) 图中仅存在孤立的结点时，即为零 DWFC 图。

性质 3 当图为零 DWFC 图时，类内部成员间依赖关系最弱，类内聚度最低。

定义 5(完全 DWFC 图) 设图中属性、方法结点数分别为 n、m，当连接边数为 $2(m^2+n^2+m^* n)$ 时即为完全 DWFC 图。

性质 4 当图为完全 DWFC 图时，类内部成员间依赖关系最强，类内聚度最高。

定义 6 当图中仅存在边{$e \in E \mid source(e)=m \cap target(e)=n \cap type(m)=method \cap type(n)=attribute$}时，即为二分 DWFC 图。

性质 5 二分 DWFC 图中仅存在方法与属性间的依赖关系，此时方法间、属性间的依赖性最低。

定义 7(特殊方法与常规方法) 对一给定的类 C，类中定义的构造函数、析构函数、访问方法和授权方法统称为特殊方法，除此之外的其它方法称为常规方法。

性质 6 特殊方法对类内聚不造成影响，在度量中不参与计算。

定义 8 设类 A 和类 B 生成的 DWFC 图分别为 G_A、G_B，若 G_A 与 G_B 同构，则称类 A 和类 B 具有同构 DWFC 图。

性质 7 若类 A 和类 B 具有同构 DWFC 图，则类 A 和类 B 具有相同的内聚性。

2.3 基于依赖关系的内聚度量定义(假设类中方法、属性数分别为 m、n)

(1)属性间的内聚度量：考虑两种属性间的依赖关系对内聚影响程度不同，我们采用加权的方法来进行度量。

$$Cohesion(A\_A)=\begin{cases}0, & n=0\\ 1, & n=1\\ (1/n)\sum_{i=1}^{n}((Waa1*NAA1i+Waa2*NAA2i)/ & \\ (Waa1+Waa2)*NAA), & n>1\end{cases}$$

$NAA1i=NC(ni, attribute, attribute, a \xrightarrow{D} b)$：表示类中属性结点 ni 直接依赖的属性结点的数目，$NAA1i \in [0, NAA]$，Waa1 表示相应权值。

$NAA2i=NC(ni, attribute, attribute, a \xrightarrow{ID} b)$：表示类中属性结点 ni 间接依赖的属性结点的数目，$NAA2i \in [0, NAA]$，Waa2 表示相应权值。

NAA：表示类中属性数目 n。

(2)方法间的内聚度量：

$$Cohesion(M\_M)=\begin{cases}0, & m=0\\ 1, & m=1\\ (1/m)\sum_{i=1}^{m}((Wmm1*NMM1i+Wmm2* & \\ NMM2i)/(Wmm1+Wmm2)*NMM), & m>1\end{cases}$$

$NMM1i=NC(ni, method, method, A \xrightarrow{IV} B)$：表示类中方法结点 ni 调用依赖的方法结点的数目，$NMM1i \in [0, NMM]$，Wmm1 表示相应权值。

$NMM2i=NC(ni, method, method, A \xrightarrow{AD} B)$：表示类中方法结点 ni 属性依赖的方法结点的数目，$NMM2i \in [0, NMM]$，Wmm2 表示相应权值。

NMM：表示类中方法数目 m。

(3)方法与属性间的内聚度量：

$$Cohesion(M\_A)=\begin{cases}0, & m=0 \vee n=0\\ (1/m)\sum_{i=1}^{m}((Ww*Nwi+Wr*Nri)/ & \\ (Ww+Wr)*NAA), & m>0 \wedge n>0\end{cases}$$

$Nwi=NC(ni, method, attribute, A \xrightarrow{WD} a)$：表示方法 i 中写依赖的属性结点数目，$Nwi \in [0, NAA]$，Ww 表示相应权值。

$Nri=NC(ni, method, attribute, A \xrightarrow{RD} a)$：表示方法 i 中读依赖的属性结点数目，$Nri \in [0, NAA]$，Wr 表示相应权值。

NAA:表示类中属性数目 n。

3 理论证明

Briand 提出了一个好的面向对象模块内聚度量应该能满足的四个性质,即非负性、最小值与最大值、单调性和模块合并后内聚不会增大,本小节将以属性间的内聚依赖 Cohesion(A_A)关系为例证明本文提出的方法能够满足这样四个性质:

(1)非负性(标准化):

1)当 n=0 即类中不涉及任何属性,Cohesion(A_A)=0;

2)当 n>0 且类中不存在任何属性直接或间接的依赖于自身或者其余任何属性时,Cohesion(A_A)=0;

3)当 n>0 且存在属性间的依赖关系时,Cohesion(A_A)>0,所以 Cohesion(A_A)度量满足非负性(标准化)。

(2)最大值与最小值:即证 Cohesion(A_A)∈[MIN,MAX]。

1)根据 Cohesion(A_A)定义,当 n=0 或 1 时,Cohesion(A_A)∈[0,1];

2)若类中每个属性都直接且间接依赖于自己以及类中其余属性,则 Cohesion(A_A)为最大值 MAX=1;

3)若类中不涉及任何属性或者类中不存在任何属性直接或间接依赖于自身或者其余任何属性,则 Cohesion(A_A)为最小值 MIN=0。因此,对于任意属性 a,(Waa1 * NAA1+Waa2 * NAA2)<=(Waa1+Waa2) * NAA,因此 Cohesion(A_A)∈[MIN,MAX]=[0,1]。

(3)单调性:即证明在 DWFC 中属性结点之间增加若干条边,内聚性不会减少。

证明:假设属性结点 a 直接和间接依赖的属性结点数目分别为 NAA1、NAA2。

在 DWFC 中以属性结点 a 为起点增加若干条边以后 a 直接和间接依赖的属性结点数目分别为 $NAA1_1$、$NAA2_1$;

显然 $NAA1_1>=NAA1$ 且 $NAA2_1>=NAA2$,(Waa1 * NAA11+Waa2 * NAA21)>=(Waa1 * NAA1+Waa2 * NAA2);

又因为增加边后属性结点总数目 NAA 不变,

$Cohesive(A\_A)_1=(Waa1 * NAA1_1+Waa2 * NAA2_1)/(Waa1+Waa2) * NAA$;

所以 $Cohesive(A\_A)_1>=Cohesive(A\_A)$,即在图中属性间添加若干边后,Cohesive(A_A)不会减小,满足单调性。

(4)模块合并后内聚不会增大:

设有两个类 C_1、C_2,属性间的内聚度分别为 $Cohesive(A\_A)_1$ 和 $Cohesive(A\_A)_2$,两个类简单合并后内聚度为 $Cohesive(A\_A)_{12}$,即证

$Cohesive(A\_A)_{12}<=MAX(Cohesive(A\_A)_1,Cohesive(A\_A)_2)$。

证明:两个类简单合并以后,类中原有属性间各种依赖关系并没有改变。

设类 C_1 和 C_2 分别有 n_1 和 n_2 个属性,考虑如下几种情况:

1)若 $n_1=n_2=0$,则 $Cohesive(A\_A)_{12}=Cohesive(A\_A)_1=Cohesive(A\_A)_2=0$;

2)若 n_1 或 $n_2=1$,不妨设 $n_1=1$,则 $Cohesive(A\_A)_1=1$,即 C_1 的属性内聚为最大值 1,根据 Briand 性质 2(最大值与最小值),合并后的属性间度量值

$Cohesive(A\_A)_{12}<=MAX=Cohesive(A\_A)_1=1$,

即证 $Cohesive(A\_A)_{12}<=MAX(Cohesive(A\_A)_1,Cohesive(A\_A)_2)=1$。

3)若 $n_1,n_2>1$,因属性间内聚度 $Cohesion(A\_A)=(1/n)\sum_{i=1}^{n}((Waa1 * NAA1i+Waa2 * NAA2i)/(Waa1+Waa2) * NAA)$,令 $m_1=\sum_{i=1}^{n1}(Waa1 * NAA1i+Waa2 * NAA2i)/(Waa1+Waa2)$,$m_2=\sum_{j=0}^{n2}(Waa1 * NAA1j+Waa2 * NAA2j)/(Waa1+Waa2)$,$m_{12}=\sum_{j=0}^{n1}(Waa1 * NAA1i+Waa2 * NAA2i)/(Waa1+Waa2)+\sum_{j=0}^{n2}(Waa1 * NAA1j+Waa2 * NAA2j)/(Waa1+Waa2)=m_1+m_2$,则 $Cohesive(A\_A)_1=m_1/n_1^2$,$Cohesive(A\_A)_2=m_2/n_2^2$,

$Cohesive(A\_A)_{12}=(m_1+m_2)/(n_1+n_2)^2$。

不妨假设 $Cohesive(A\_A)_1>=Cohesive(A\_A)_2$,即证 $Cohesive(A\_A)_{12}<=Cohesive(A\_A)_1$。

证明:$Cohesive(A\_A)_{12}=(m_1+m_2)/(n_1+n_2)^2$,

$Cohesive(A\_A)_{12}-Cohesive(A\_A)_1=(m_1+m_2)/(n_1+n_2)^2-(m_1/n_1^2)=(m_2n_1^2-m_1n_2^2-2m_1n_1n_2)/(n_1+n_2)^2n_1^2$,

又因 $Cohesive(A\_A)_1>=Cohesive(A\_A)_2$,$m_1/n_1^2>=m_2/n_2^2$,

则 $n_2^2 * m_1>=n_1^2 * m_2$,$(m_2n_1^2-m_1n_2^2-2m_1n_1n_2)<=0$,$Cohesive(A\_A)_{12}-Cohesive(A\_A)_1<=0$,

所以 $Cohesive(A\_A)_{12}<=Cohesive(A\_A)_1$,$Cohesive(A\_A)_{12}<=MAX(Cohesive(A\_A)_1,Cohesive(A\_A)_2)$。

即证,类 C_1、C_2 合并后,属性间内聚度不会增加。

综上,本文提出的内聚度量方法中属性间的内聚度量 Cohesive(A_A)满足一个良好内聚度量方法的四个性质。同理可得方法间内聚度量以及属性与方法间的内聚度量均满足这样四个性质,因此由 DWFC 模型构造的该度量方法是一个经过 Briand 理论证明的良好度量准则。

4 内聚度量验证规则改进

由于 Briand 提出的四个性质仅是一个良好度量准则所必须满足的必要条件而缺乏充分性,满足这样的条件并不能够完全保证一个度量准则与人的直觉相符合,所以我们通过利用 PROBE 方法来分析一个优良度量准则与人的直觉判定间的相关性以及有效性,从而进一步完善理论验证度量准则正确性以及合理性的必要条件。

个人软件过程(Personal Software Process,简称 PSP)是一个自我改进的软件开发过程,用来帮助人们控制和改进工作方式。PSP 能够提供所需要的历史性数据,辅助开发人员预计可能的变更以及有效的履行承诺。PSP 还能对工作进行测度,有助于我们从效率波动中进行学习,然后把这些知识集成到一个不断增长的文档化实践中。

区别于传统的软件规模估算方法,PSP 提出一种基于代理的规模估算方法,能将软件产品的规模和估算者能想象和描述的功能联系起来。代理是一种替代品,帮助我们更形象地进行估算,更容易判断产品的规模。根据具体估算软件的特点,代理可以有多种选择。由于面向对象的开发日益广泛,

对象是一个很能逼近开发特征的概念，因此选用对象作为代理就得到了 PROBE 方法，即基于代理的估算(PROxy-Based Estimating)。估算流程如图 3 所示。

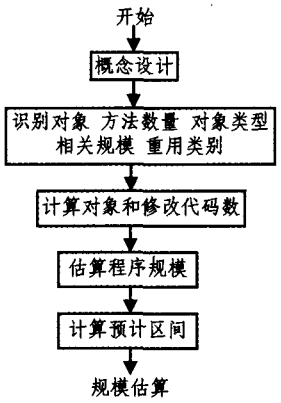

图 3 基于代理的估算流程图

4.1 利用相关性与有效性进行内聚度量验证

对于现有的度量验证方法，我们可以采用 Briand 提出的四个准则进行必要性验证或者采用实例数据进行实验证明。但是满足 Briand 性质并不一定保证该准则的合理性，所以应当将人的直觉判断与检验度量准则相结合，把是否与人的直觉相符合作为判断度量准则合理性的必要条件之一，由此本文将 Briand 必要条件扩充为五条：即非负性、最大值与最小值、单调性、模块合并内聚度不会增加以及度量与直觉相符合。而 PROBE 方法采用了相关性以及有效性来验证两组数据间连接的紧密性，我们将用此方法验证度量准则是否与人直觉相符合。

4.1.1 相关性分析

PROBE 方法采用分析开发时间与程序复杂度间的相关性来证明用程序复杂度度量开发时间是科学可行的。同理，我们采用该方法来分析一个内聚度量准则与人的直觉对类内聚的判断之间的相关性来验证提出的度量准则是否合理，从而验证能否用该度量准则刻画类内聚性。

相关性分析过程定义如下：

Step1：数据准备(选取 n 个类以供相关性分析)。

Step2：选取待验证的度量准则对 n 个类进行内聚度量，令 x_i 为第 i 个程序内聚度量结果(假定选取的内聚度量准则已经满足 Briand 性质，即 $x_i \in [0,1]$)。

Step3：采用 Wideband-Delphi 方法集成多个专家意见，令 y_i 为多个专家根据经验给出的第 i 个程序的内聚度量结果，$y_i \in [0,1]$。

Step4：对 x_i 与 y_i 进行相关性分析，令结果为

$$r(x,y)=\frac{n\sum_{i=1}^{n}x_iy_i-\sum_{i=1}^{n}x_i\sum_{i=1}^{n}y_i}{\sqrt{[n\sum_{i=1}^{n}x_i^2-(\sum_{i=1}^{n}x_i)^2][n\sum_{i=1}^{n}y_i^2-(\sum_{i=1}^{n}y_i)^2]}}$$

Step5：结果分析：

1)当 $0.9<=r^2$ 时，$f(x)$ 与 $g(x)$ 相关性很强，数据集间几乎是相互依赖；

2)当 $0.7<=r^2<0.9$ 时，相关性较强；

3)当 $0.5<=r^2<0.7$ 时，相关性一般；

4)当 $r^2<0.5$ 时，相关性较弱。

当两组数据集相关性达到较强时，采用该度量准则进行内聚度量与人的直觉相符合，满足度量准则符合直觉判断的必要条件，由此证明采用该方法进行内聚度量的可行性。

4.1.2 对相关性的有效性进行分析

进行相关性分析的数据项多少直接影响了相关的偶然性，即在数据项极少的情况下产生相关性的偶然因素是很大的，所以即便有高的相关系数也不能证明两者间必然相关，我们有必要在此基础上进行相关的显著性(有效性)分析。

有效性分析过程如下：

Step1：数据准备(获取数据项数目 n 以及相关系数 r)。

Step2：进行有效性分析：

1)计算 t 分布值，$t=\frac{|r(x,y)|\sqrt{n-2}}{\sqrt{1-r(x,y)^2}}$；

2)在“t 分布表”中自由度为 $n-2$ 那行，按双侧置信区间查找 1)中 t 值的概率 p；

3)对自由度为 $n-2$ 的 t 分布从负无穷到 t 进行积分，得到概率 p；

4)计算两个尾部的值，令 $e=2^*(1-p)$。

Step3：结果分析：

1)$e<=0.05$，说明偶然发现相关性的可能性较小，可以认为数据集间有较强相关性；

2)$e>0.20$，说明偶然发现相关性的可能性较大，这可以认为没有较强相关性；

3)$0.05<e<=0.20$，有效性介于二者间。

当参与分析的数据集较少的情况下，该方法能够充分验证数据项间相关性的有效程度。

4.2 使用改进的验证规则对比验证已有度量方法

本文采用 java 语言实现了一个经典的电梯问题程序，选取其中 10 个类结合多种度量方法进行内聚度量，得到如下的度量结果对比表。

表 1 各类方法度量结果

	专家结论	DWFC	Briand'sRCI	Chen-Xu's
C1	0.50	0.47	0.56	0.29
C2	0.70	0.73	1.00	0.51
C3	0.20	0.04	0.00	0.03
C4	0.80	0.82	1.00	0.40
C5	0.30	0.30	0.85	0.50
C6	0.50	0.50	0.25	0.70
C7	0.80	0.75	0.30	1.00
C8	0.20	0.23	0.80	0.40
C9	0.80	0.78	0.70	1.00
C10	0.20	0.19	0.50	0.40

表 1 中“专家结论”数据采用 Wideband-Delphi 方法结合多名具有丰富经验的编程人员以及从事面向对象研究的专家度量得到，利用 SPSS 软件分别计算 DWFC、Briand'sRCI、Chen-Xu's 三种已经满足 Briand 准则的方法得到的度量结果与“专家结论”间的相关性分别为 r1，r2，r3，结果如下表 2 所示。

表 2 相关性分析结果

	DWFC	Briand'sRCI	Chen-Xu's
r	r1=0.977	r2=0.404	r3=0.385
r^2	$r1^2$=0.954	$r2^2$=0.164	$r3^2$=0.148

实验结果显示，DWFC 方法与专家结论间相关性 $r^2>0.9$，即两者间具有强相关性，满足新的验证规则对符合人的

直觉判断的要求，是一个良好的度量准则。同时也验证了新的度量验证准则能够有效的判断度量方法是否符合人的直觉判定。

由于数据量相对较小，我们以 r1 为例对相关系数进行有效性分析得到：

t1=9.259

观察 t 分布表中自由度为 8 的那行，表中最大值是 3.355，概率为 0.99，所以 p>0.99。因此偶然得到 r1 这个相关系数的概率为 2*(1−p)，不到 0.02。所以 DWFC 方法与专家判定结果间的相关性是高度有效的，可以利用 DWFC 方法来进行内聚性度量，其结果符合人的直觉判定。

通过以上的验证规则，我们改进了现有的理论验证方法，在 Briand 的基础上结合人的直觉进行判断，并提出了具体的方法来完成该验证过程。

5 实例分析

我们以一个简化的 JAVA 中定义的类的例子来加以讨论，比较本文提出的度量方法较之现有方法之间的改进。

```
  Class Triple{
float temp,temp1,temp2;
Triple (float a,floatb,floatc)
{temp=a;temp1=b;temp2=c;}
Triple (float a,float b){temp=a;temp1=b;}
public float sin(float x)
{temp=temp+1;return temp;}
public float cos(float x)
{temp=temp+1;return temp;}
public float tg(floatx)
{temp1=sin(x);temp2=cos(x);temp=temp1/temp2;return temp;}
public float ctg(float x)
{temp1=sin(x);temp2=cos(x);temp=temp2/temp1;return temp;}
public float self(){if (temp1<0) temp=2;else {temp1=temp1-1;
temp=temp*self();}
return temp;}
};
```

本例定义了类 Triple，其中成员方法以及成员变量集合分别为：3 个属性{temp，temp1，temp2}以及 7 个方法{Triple，Triple，sin，cos，tg，ctg，self}。其中两个 Triple 方法为该类的构造方法属于特殊方法，在参与度量前被排除，所以类中参与计算的方法数仅为 5 个。根据本文提出的方法的定义，得到关于类内聚的三个度量值的计算过程如下（对于权值的计算可以根据历史数据通过统计方法得到合适的值，为简化计算本文将权值根据经验简单设置为 Waa1=Wmm1=Ww=2，Waa2=Wmm2=Wr=1）：

Cohesion(A_A)=(1/3)(8/9+2/9+0)=10/27；

Cohesion(M_A)=(1/5)(3/9+3/9+8/9+8/9+6/9)
=28/45；

Cohesion(M_A)=(1/5)(5/15+5/15+7/15+7/15+7/15)=31/75；

将类 Triple 中属性 temp1、temp2 作为相应方法的局部变量后得类 Triple1，而将类 Triple 中属性 temp、temp1、temp2 均作为相应方法的局部变量后得类 Triple2，由此得到下列表 3。

表 3 不同度量方法内聚度比较

Class	LCOM	Briand'sRCI	Chen-Xu's	DWFC
Triple	0	5/9	2/7	949/2025
Triple1	0	1	32/63	11/15
Triple2	21	0	2/63	2/45

表 3 给出了四个不同的度量方法对 Triple 、Triple1、Triple2 三个类度量的结果，由此可以看出：

(1)从特殊方法方面考虑：LCOM、Chen-Xu's、RCI 三种方法都缺乏对特殊方法的考虑或者考虑不全面，因此度量结果会产生或高或低的偏移，造成度量结果不准确。而 DWFC 方法则在计算时排除了四种对内聚不造成影响的度量方法。例如将 DWFC 方法与同样基于依赖分析的 Chen-Xu's 方法进行比较发现后者对三个类的度量结果均小于前者，说明后者将特殊方法参与度量导致度量结果偏低。

(2)从不同联结强度考虑：LCOM、Chen-Xu's、RCI 三种方法对不同的联结强度没有进行区分，一概而论，造成度量结果不够精确。而 DWFC 方法在定义时区分了不同的联结强度并提出加权思想，能够使度量结果更加准确。例如对类 Triple 的度量，DWFC 方法的度量结果就更为精确。

(3)从是否满足理论验证考虑：LCOM 不满足 Briand 的四个基本属性，是非标准以及非单调的，度量值越小内聚度越高。

(4)从交互模式考虑：LCOM、Briand's RCI 都忽略了一些重要的交互模式，如属性间的依赖以及方法间的调用，在有些情况下易造成度量结果与直觉相悖。例如 LCOM 不能区别一些相似的类如 Triple 和 Triple1，而 Briand's RCI 对 Triple2 的度量结果为 0，没有考虑方法间的调用。

结束语 本文利用图理论，提出一种独立于具体实现语言的基于类内部方法、属性间依赖性分析的内聚度量模型。该模型采用有向带权伪图 DWFC 来客观定义类内部属性以及一般方法间的联结关系，赋予权值使得度量能够区别不同的联结关系对内聚的不同影响。而伪图包含多重边以及环，能更好地表达结点的多重关系以及结点对自身的依赖。有效避免了现有度量集在对特殊方法的考虑、忽略方法属性对自身的依赖以及是否符合良好度量准则的四个性质等方面的问题。

针对现有度量验证方法存在的问题，本文同时提出应结合人的直觉判断来进行内聚度量验证，并给出了具体的实现方法，最后通过实验验证了该方法的可行性。

参 考 文 献

1 Chidamber R, Kemerer F. A Metrics Suite for Object-Oriented Design. IEEE Trans. ,Softw. ,Eng. ,1994,20(6)

2 Zhou Yuming, Lu iangtao, Lu Hongmin, Xu Baowen. A Comparative Study of Graph Theory-based Class Cohesion Measures. Software Engineering Notes, 2004, 29(2)

3 Mens T, Lanza M. A Graph-Based Matamodel for Object-Oriented Software Metrics. Electronic Notes in Theoretical Computer Science, 2002, 72(2)

4 Vivanco R A, Pizzi N J. Identifying Effective Sofeware Metrics Using Genetic Algorithms. CCECE 2003-CCGEI, 2003, 8(3)

5 Demko A B, Pizzi N J. The Utility of Graph Theoretic Software Metrics: A Case Study. CCECE 2003-CCGEI, 2003, 8(3)

6 Wailinshaw N. The Java System Dependence Graph: [Technical-Report: EFoCS-46-2003]. 24th April 2003

7 Fioravanti F, Nesi P. A method and tool for assessing object-oriented projects and metrics management. The Journal of Systems and Software, 2000, 53: 111～136