一种基于依赖性分析的类内聚度度量方法*

陈振强[1,2], 徐宝文[1,2,3+]

[1](东南大学 计算机科学与工程系,江苏 南京 210096)

[2](江苏省 软件质量研究所,江苏 南京 210096)

[3](武汉大学 软件工程国家重点实验室,湖北 武汉 430072)

An Approach to Measurement of Class Cohesion Based on Dependence Analysis

CHEN Zhen-Qiang[1,2], XU Bao-Wen[1,2,3+]

[1](Department of Computer Science and Engineering, Southeast University, Nanjing 210096, China)

[2](Jiangsu Institute of Software Quality, Nanjing 210096, China)

[3](State Key Laboratory of Software Engineering, Wuhan University, Wuhan 430072, China)

+ Corresponding author: Phn: 86-25-3793977 ext 1, Fax: 86-25-3689779, E-mail: bwxu@seu.edu.cn

http://cse.seu.edu.cn/people/bwxu

Received 2002-09-25; Accepted 2003-06-19

Chen ZQ, Xu BW. An approach to measurement of class cohesion based on dependence analysis. *Journal of Software*, 2003,14(11):1849~1856.

http://www.jos.org.cn/1000-9845/14/1849.htm

Abstract: Cohesion represents the tight degree of the components of a software module. In an OO environment, the cohesion is mainly about the tight degree among the attributes and methods in classes. So it is urgent to fully explore the tight degree among the attributes and methods in classes of the detailed applications in order to make up the deficiency of the existing methods. The relationships between attributes and attributes, attributes and methods, methods and methods of a class, and the properties of these relationships are discussed. According to these properties, a new approach is proposed to measure the cohesion of a class based on a dependence analysis. This method is proved to satisfy the properties that a good measurement should have. The approach overcomes the limitations of the previous class cohesion measures, which consider only one or two of the three relationships in a class. The result of the approach provides a guideline for evaluating the cohesion of a class.

* Supported by the National Natural Science Foundation of China under Grant No.60073012 (国家自然科学基金); the National Grand Fundamental Research 973 Program of China under Grant No.2002CB312000 (国家重点基础研究发展规划(973)); the Program for Cross-Century Outstanding Teachers of the Ministry of Education of China (教育部跨世纪优秀人才培养计划); the National Research Foundation for the Doctoral Program of Higher Education of China under Grant No.20020286004 (高等学校博士学科点专项科研基金); the Natural Science Foundation of Jiangsu of China under Grant No.BK2001004 (江苏省自然科学基金); the Jiangsu Key Science and Technology Project of China under Grant No.BE2001025 (江苏省科技攻关项目); the Opening Foundation of State Key Laboratory of Software Engineering in Wuhan University of China (武汉大学软件工程国家重点实验室开放基金); the Opening Foundation of Jiangsu Key Laboratory of Computer Information Processing Technology in Soochow University of China (江苏省计算机信息处理技术重点实验室开放基金(苏州大学))

第一作者简介: 陈振强(1976－),男,山东新泰人,博士,主要研究领域为程序分析、理解与测试.

Key words: software metrics; object orientation; program analysis; cohesion

摘 要: 内聚度是指模块内各成分之间的联结强度.在面向对象程序中,内聚度主要是指类内部各成分之间的联结强度.在深入剖析了类的属性与属性、方法与属性以及方法与方法之间的关系及其性质的基础上,提出了一种基于依赖性分析的内聚度度量方法,并证明了它满足优良的内聚度度量方法应该具有的性质,从而克服了已有方法只是分析了某个方面的不足,为综合评价类内聚度提供了指南.

关键词: 软件度量;面向对象;程序分析;内聚度

中图法分类号: TP311 **文献标识码**: A

内聚度是指模块内部各成分之间的联结强度.内聚度越高,越容易理解、修改和维护.但内聚度本身是主观的、非形式化的概念,程序设计人员很难客观地评估一个模块的内聚度.为此,人们开发出许多度量准则用于量化模块的内聚度[1~3],为程序设计人员开发出高内聚度的模块提供指南.随着面向对象技术在系统开发中的广泛应用,面向对象程序的内聚度度量方法已成为一个重要的研究课题.

在面向对象程序设计语言中,类是支持数据抽象、封装和继承等面向对象特征的基本概念,它定义了表示状态的属性集和表示行为的方法集.因此,在面向对象程序中,内聚度主要是指一个类或一个对象内部属性和方法之间的联结强度.目前,国内外在类内聚度方面已做了大量的研究工作,并提出了一些度量准则和方法[4~7].

1 相关研究分析

Briand 指出[2,3],一个优良的模块内聚度度量准则应满足非负性及标准化、最小值与最大值、单调性和模块合并后内聚度不会增大这 4 个性质.这些性质可以用来检查现有内聚度度量准则中可能的问题,也为研究人员开发优良的内聚度度量准则提供了指南.本节根据这个准则对现有的一些方法进行简单的讨论和比较.设类 C 中共有 m 个方法和 n 个属性.Chidamber 提出了一种利用相似方法与非相似方法的差异计算内聚度的方法,称为 LCOM1.它是一种类内聚度的反向度量,其值越大,类内聚度越低.Hitz 的 LCOM2 方法用连通子图的个数来度量类的内聚度.如果 $LCOM2=1$,Hitz 引入连通度(connectivity)来区分不同的度量.Henderson 的 LCOM3 度量方法可描述为

$$LCOM3(C)=\frac{\frac{1}{n}\sum_{j=1}^{n}|\mu(a_j|-m)}{1-m},$$

其中 $\mu(a)=\{M|a\in M\_A(M)\}$,a 是属性,M 是方法,$M_A(M)$表示方法 M 中引用的属性集.

Briand 的 RCI 度量是用交互图中的边数和属性与方法间可能的最大交互数的比例.Chae 的胶水方法(glue method)、Xu-Zhou 的胶水边(glue edges)分析了交互图中的交互模式,因此其度量更加合理[8].

(1) 所有这些方法仅考虑了方法对属性的简单引用关系.

(2) LCOM1,LCOM2 和 LCOM3 是非标准的.LCOM1 是非单调的.

(3) RCI 满足 Briand 的 4 个基本属性,但是没有考虑交互模式;LCOM1,LCOM2 和 LCOM3 也没有考虑交互模式.因此,有些情况下它们的度量与直觉不相符.

(4) Chae 的 CO 克服了以前方法的大部分局限,但它是非单调的[8].Xu-Zhou 的 CO 是对 Chae 方法的改进.这两种方法最大的缺点是它们只能应用于连通图,对非连通图其结果恒等于 0.

(5) LCOM1 和 LCOM2 只度量了类中方法间的内聚度.

(6) LCOM3,Chae 和 Xu-Zhou 的 CO 只度量了类中方法与属性间的内聚度.

为了弥补现有方法的不足,我们曾对类内聚度度量做过比较深入的研究,取得了一定的成果[8,9].但这些方法主要针对类本身的特性,并没有考虑具体应用系统.

为了充分探明在具体应用系统中类的方法及属性间的紧密程度,本文在已有工作的基础上,对类中属性与属性间、方法与方法间以及方法与属性间的各种关系进行了深入剖析,在综合分析这 3 方面关系的基础上,提

出了一种比较全面的内聚度度量方法,并证明了此方法满足一个优良的模块内聚度度量准则所应满足的 4 个性质,从而克服了现有许多类内聚度度量准则仅考虑某个方面而没有综合考虑的不足.

2 基本概念和性质

本文将属性和方法之间的 3 种关系定义为属性间、方法间以及方法对属性等 3 种依赖关系.

定义 1. 在类的方法中,若对属性 a 的定义(值改变)直接或间接引用(值不变)了 b,或者属性 a 能否被定义由 b 的状态决定,则称 a 依赖于 b,记为 $a\rightarrow b$.

属性 a 能否被定义由 b 的状态决定一般发生在控制结构中(如 if,while),即 b 为控制条件中引用的属性,而定义 a 的语句在控制结构内部.

定义 2. 若方法 M 中使用了属性 a,则称 M 依赖于 a,记为 $M\Rightarrow a$.

定义 3. 方法间的依赖关系有两种:调用关系和数据依赖关系.若方法 M 中调用了方法 P,则称 M 调用依赖于 P,记为 $M\rightarrow P$;若方法 M 引用的属性 a 由方法 P 定义(修改),则称 M 中引用的属性 a 依赖于 P 中定义的属性 a,记为 $M\xrightarrow{a,a}P$.为了统一,对调用边也添加一个标记(*,*),即若 $P\rightarrow Q$,则 $P\xrightarrow{*,*}Q$.

在不引起混淆的情况下,下文中方法间的依赖关系是指数据依赖关系.

为了计算这种依赖关系,对每个方法 M 引入 3 个集合:OUT,DEP_A 和 DEP_A_OUT.DEP_A 和 DEP_A_OUT 的元素形式为$\langle a,b\rangle$,其中,a 和 b 为类的属性.

• $OUT(M)$表示在方法 M 中值被修改的属性集.

• $DEP_A(M)$表示 M 中引用的属性对在 M 外定义属性的依赖集;$DEP_A(M)$是属性间依赖关系的子集,它只记录 M 中引用的属性对其他方法定义属性的依赖关系.

• $DEP_A_OUT(M)$表示当 M 返回时,M 中引用的属性对外部定义属性的依赖集.

方法中计算得到的中间结果通常对外是不可见的,且属性在一个方法中可以被多次赋值.为了提高分析的精度,引入集合 DEP_A_OUT 来记录方法返回时属性间的依赖关系.显然有 $DEP\_A\_OUT(M)\subseteq DEP\_A(M)$.

性质 1. 方法间的调用关系具有传递性,即 $M\rightarrow P,P\rightarrow Q\Rightarrow M\rightarrow Q$.

定义 4. 设 $SS=\langle s_1,s_2,\ldots,s_n\rangle$为一个语句序列,若 s_i 执行完后可能立即执行 $s_{i+1}(i=1,2,\ldots,n-1)$,则称 SS 为一个可执行路径.

性质 2. 设 $SS=\langle s_1,s_2,\ldots,s_n\rangle$为一个可执行路径,若下面 3 个条件都满足,则 $M\xrightarrow{a,b}P$.

(1) s_1 调用了对象 O 的方法 P,s_n 调用了 O 的方法 M;

(2) $\langle a,b\rangle\in DEP\_A(M),b\in OUT(P)$;

(3) 在$\langle s_2,\ldots,s_{n-1}\rangle$上没有调用 O 的方法或者对任意调用的方法 Q,$b\notin OUT(Q)$.

根据性质 2,我们定义方法间的传递依赖如下:

定义 5. 设 $SS=\langle s_1,s_2,\ldots,s_n\rangle$为一个可执行路径,在 SS 上依次调用了对象 O 的方法 $M_1,\ldots,M_m(m>1)$,则 M_m 传递依赖于 M_1,记为 $M_m\xrightarrow{a,c}M_1$,当且仅当如下两个条件都成立:

(1) $\exists a,c(M_m\xrightarrow{a,c}M_1)$;

(2) $M_m\xrightarrow{a,b}M_i\wedge\exists\langle b,c\rangle(\langle b,c\rangle\in DEP\_A\_OUT(M_i),c\in OUT(M_1))$.

在不需要区分属性的情况下,M_m 传递依赖于 M_1,可简记为 $M_m\xrightarrow{*}M_1$.

为统一起见,我们将类的各种依赖关系用类依赖图的形式来描述.

定义 6. 一个类 C 的依赖图(CDG)是有向图,$CDG=\langle N,E,T\rangle$,其中 N 是节点集,E 是边集,T 是标记集.$N=N_A\cup N_M$,N_A 是属性节点集,每个节点表示一个属性;N_M 是方法节点集,每个节点表示一个方法.CDG 由 3 个子图组成:属性间依赖子图(AAG)、方法间依赖子图(MMG)和方法与属性间的依赖子图(MAG).其中:

• $AAG=\langle N_A,E_A\rangle$,其中 N_A 是属性节点集;E_A 是边集,如果 $a\rightarrow b$,那么边$\langle a,b\rangle\in E$.

• $MMG=\langle N_M,E_M,T\rangle$,其中 N_M 是方法节点集;E_M 是边集,表示方法间的依赖关系;$T\in E\times(V,V)$是标记集,其中 V 是属性与{*}的并集.

• $MAG=\langle N,E_{MA}\rangle$,其中 N 是节点集;E_{MA} 是边集,表示方法对属性的依赖关系.MAG 中所有的边都由方法节点

指向属性节点,因此 MAG 是一个二部图.

各种依赖关系及类依赖图可以通过分析程序的控制流、数据流获得[10,11],限于篇幅,本文不再详述.

3 类内聚度度量方法

如上所述,类由属性和方法组成,属性和方法间存在 3 种关系,因此,类的内聚度也应从这 3 个方面进行度量.设类 C 中共有 n 个属性,m 个方法,其中 $m,n\geq 0$.

3.1 属性间的内聚度度量

属性间的内聚度主要是指各属性之间的关系(本文系指依赖关系)的密切程度.为了度量这种关系,对每个属性 a,引入一个集合 A_DEP 来记录 a 所依赖的其他属性的集合:

$$A\_DEP(a)=\{b|a\rightarrow b,a\neq b\}.$$

如果每个属性都依赖于类中其他所有属性,那么它们之间的关系最密切.因此,我们定义属性间的内聚度为

$$Cohesion(A\_A)=\begin{cases}0, & n=0\\ 1, & n=1\\ 1/n\sum_{i=1}^{n}\dfrac{|A\_DEP(a_i)|}{n-1}, & n>1\end{cases},$$

其中 $\dfrac{|A\_DEP(a)|}{n-1}$ 是属性 a 依赖变量的个数与其他属性总数的比例,表示 a 对类中其他属性的依赖程度.若 n=0,则类中没有属性,此时属性间的内聚度为 0;若 n=1,虽然这个属性不会依赖于其他属性,但由于类中只有这个属性,它自身是最紧密的,因此内聚度为 1.

3.2 方法与属性间内聚度度量

方法与属性间的关系是已往分析的重点,目前已有许多良好的计算方法,如 Chae 的方法[5,12].这些方法大都是基于 MAG 上的图论方法.如上所述,这些方法只是简单描述了方法对属性的引用关系,但方法之间是否真正存在关系,并没有准确描述.因此,我们需要对这些方法加以改进.为了描述上的完整性,我们用 Cohesion (previous)来表示一个已有的、具有优良特性的内聚度度量方法.

对每个方法 M,我们引入两个集合:M_A 和 M_A_OUT.其中:

- M_A 记录一个方法中使用的属性集.
- M_A_OUT 记录一个方法中出现的与其他方法有关系的属性集,即

$$M\_A\_OUT(M)=\{a|\exists b,P(P\xrightarrow{b,a}M\vee M\xrightarrow{a,b}P)\wedge a,b\neq\text{`*'}\}.$$

显然有 $M\_A\_OUT(M)\subseteq M\_A(M)$.这样,我们定义方法与属性间的内聚度为

$$Cohesion(M\_A)=\begin{cases}0, & m=0\vee n=0\\ \dfrac{|M\_A(M)|}{n}, & m=1\\ 1/m\sum_{i=1}^{m}Cohesion(Previous)*\dfrac{|M\_A\_OUT(M_i)|}{|M\_A(M_i)|}, & \text{其他}\end{cases},$$

其中 $\dfrac{|M\_A\_OUT(M)|}{|M\_A(M)|}$ 是 M 中引用的与其他方法有关的属性与所引用的全部属性的比例,简记为 $\rho(M)$.若 $M\_A(M)=\varnothing$ 即 M 没有引用任何属性,则 $\rho(M)$ 为 0.

若 m=0 或 n=0,则方法不会引用属性,因此其内聚度为 0;若 m=1,其内聚度由其引用的属性数目来决定.

3.3 方法间内聚度度量

虽然不同方法可以通过属性连接起来,但是这并不表示在这些方法之间存在某些关系.如果它们之间存在某些关系,那么它们的紧密程度如何也需要我们进一步判断.这个过程称为方法间的内聚度度量.

对每个方法 M,我们再引入一个集合 M_DEP.

M_DEP 记录一个方法所依赖的其他方法集,

$$M\_DEP(M)=\{P|M \xrightarrow{*} P\}.$$

这样,方法间的内聚度 $Cohesion(M_M)$可通过如下方法计算:

$$Cohesion(M\_M)=\begin{cases}0, & m=0\\ 1, & m=1\\ 1/m\sum_{i=1}^{m}\dfrac{|M\_DEP(M_i)|}{m-1}, & m>1\end{cases},$$

其中 $\dfrac{|M\_DEP(M)|}{m-1}$ 表示方法 M 与类中其他方法的关系,取值越大表示与 M 有关的方法越多.

3.4 类内聚度度量及实例分析

经过上面讨论的 3 种独立的度量后,我们可用一个三元组来表示类的内聚度,每个度量作为一个域:

$$Cohesion(C)=\langle Cohesion(A\_A),Cohesion(M\_A),Cohesion(M\_M)\rangle.$$

为了便于讨论,我们以一个典型的 C++类的例子加以说明.该类定义如下(记作 Class1):

```
class Tri {
    float temp, temp1, temp2;
public:
    float sin (float x) {temp=…; return temp;}
    float cos (float x) {temp=…; return temp;}
    float tg (float x) {temp1=sin(x);temp2=cos(x); temp=temp1/temp2; return temp;}
    float ctg (float x) {temp1=sin(x);temp2=cos(x);temp=temp2/temp1; return temp;}
};
```

所定义的类叫做 Tri,它有 3 个属性(temp,temp1 和 temp2)与 4 个方法(sin,cos,tg 和 ctg).根据定义,

$$OUT(\text{sin})=OUT(\text{cos})=\{\text{temp}\};OUT(\text{tg})=OUT(\text{ctg})=\{\text{temp,temp1,temp2}\}.$$

由方法 tg 或 ctg 可得:temp→temp1;temp→temp2.因此,

$$A\_DEP(\text{temp})=\{\text{temp1,temp2}\};\text{tg}\rightarrow\text{sin};\text{tg}\rightarrow\text{cos};\text{ctg}\rightarrow\text{sin};\text{ctg}\rightarrow\text{cos}.$$

没有方法引用其他方法定义对变量.因此,所有方法的 DEP_A,DEP_A_OUT 和 M_A_OUT 都为空集.

$$M\_A(\text{sin})=M\_A(\text{cos})=\{\text{temp}\};M\_A(\text{tg})=M\_A(\text{ctg})=\{\text{temp,temp1,temp2}\}.$$

$$M\_DEP(\text{sin})=M\_DEP(\text{cos})=\varnothing;M\_DEP(\text{tg})=M\_DEP(\text{ctg})=\{\text{sin,cos}\}.$$

这样,Tri 的内聚度为

$$Cohesion(A\_A)=1/3;Cohesion(M\_A)=0;Cohesion(M\_M)=1/3.$$

表 1 给出了不同方法的度量结果.表中 Class2 是将 Class1 的 temp1 和 temp2 作为 tg 和 ctg 局部变量得到的例子.Class3 是将 Class1 的 temp,temp1 和 temp2 作为局部变量得到的例子.在 LCOM2 那一列,括号中的数指的是连接度.我们的方法的结果是 3 个域的平均值.从表 1 可以看出如下几点:

(1) LCOM1 和 LCOM2 无法区分 Class1 和 Class2.

(2) LCOM3,Chae,Briand 和 Xu-Zhou 的度量没有考虑方法间的调用关系.

(3) LCOM3,Chae,Briand 和 Xu-Zhou 的方法度量 Class2 的结果为最大值 1.但是,如果不考虑调用关系,这些方法间根本互不相干.

Table 1 Class cohesions of different measures

表 1 不同度量的内聚度

Class	Briand's	Chae's	LCOM1	LCOM2	LCOM3	Xu-Zhou's	Our's method
Class1	2/3	1/2	0	1(1)	4/9	1/12	2/9
Class2	1	1	0	1(1)	1	1	4/9
Class3	0	0	6	4	0	0	1/9

4 性质证明

本节将证明本文提出的方法(以下简称 CH 度量)满足 Briand 的 4 个性质[2,3],即非负性及标准化、最小值和最大值、单调性和模块合并后内聚度不会增大.

根据定义,当 n=0 或 1 时,$CH(A\_A)\in[0,1]$.显然,若类中每个属性都与其他属性有关,则 $CH(A_A)$为 1;若所有属性都可单独存在,即不依赖于其他任何属性,则 $CH(A_A)$为 0;对任意属性 a,$|A\_DEP(a)|\le n-1$,因此,$CH(A\_A)\in[0,1]$.同理可知,$CH(M\_M)\in[0,1]$.

显然,如果类中没有属性或方法,那么方法与属性间就不存在依赖关系,M_A 为空集,$|M\_A(M)|=0$,因此 $CH(M_A)$为 0.如果方法 M 访问的每个属性都与其他方法有关,那么$\rho(M)$为 1.由于$|M_A(M)|\le n$,$M\_A(M)\supseteq M\_A\_OUT(M)$,所以,$CH(M\_A)\le 1$.

定理 1. 在 AAG 中添加一条边,$CH(A_A)$不会减小.

证明:设添加边之前 a 所依赖的其他属性的集合为 $A\_DEP(a)$,添加一条边$\langle a,b\rangle$后的集合为 $A\_DEP'(a)$,

a) 如果 $b\in A\_DEP(a)$,由 A_DEP 的定义可知,a 通过某些属性传递依赖于 b,因此添加边$\langle a,b\rangle$后,$A_DEP(a)$不会改变,即 $A\_DEP(a)=A\_DEP'(a)$,根据集合 A_DEP 的定义,其他属性所依赖的属性集合也不会改变.根据计算公式,$CH(A_A)$保持不变;

b) 如果 $b\notin A\_DEP(a)$,由 A_DEP 的定义可知,添加边$\langle a,b\rangle$后,$A_DEP'(a)$会增大,$A\_DEP'(a)=A\_DEP'(a)\cup\{b\}$.根据集合 A_DEP 的定义,其他属性所依赖的属性集合不会变小.根据计算公式,$CH(A_A)$增大.

总之,在 AAG 中添加一条边,$CH(A_A)$不会减小. □

Cohesion(previous)是我们直接引用的已有方法,对它的性质我们不作详细讨论,但我们假设它满足 Briand 的 4 个性质.因此,对 $CH(M\_A)$,我们主要讨论$\rho(M)$的有关性质.由 M_A_OUT 的定义可知,$\rho(M)$可通过分析 MMG 获得.

定理 2. 在 MMG 中添加一条边 $M\xrightarrow{a,b}P$,$\rho(M)$和$\rho(P)$不会减小.

证明:设添加边之前 M 使用的属性集为 $M\_A(M)$,其中与其他方法有关系的属性集为 $M\_A\_OUT(M)$,加边后这两个集合分别为 $M_A'(M)$和 $M\_A\_OUT'(M)$,

a) 如果 $a\in M\_A\_OUT(M)$,根据定义,加边后两个集合保持不变,因此,$\rho(M)$不变.

b) 如果 $a\notin M\_A\_OUT(M)$,但 $a\in M\_A(M)$,加边后 $M_A(M)$保持不变,但 M_A_OUT 会增大,即 $M\_A\_OUT'(M)=M\_A\_OUT(M)\cup\{a\}$,因此,$\rho(M)$会增大.

c) 如果 $a\notin M\_A(M)$,由于 $M\_A\_OUT(M)\subseteq M\_A(M)$,$a\notin M\_A\_OUT(M)$,因此,加边后两个集合都增大,

$$M\_A\_OUT'(M)=M\_A\_OUT(M)\cup\{a\},M\_A'(M)=M\_A(M)\cup\{a\},$$

$$M\_A\_OUT'(M)=M\_A\_OUT(M)+1,M\_A'(M)=|M\_A(M)|+1,$$

$$\frac{|M\_A\_OUT(M)|}{|M\_A(M)|}\le\frac{|M\_A\_OUT(M)|+1}{|M\_A(M)|+1},$$

$\rho(M)$不会变小.

d) 如果添加的边为调用边,那么

$$M\_A\_OUT'(M)=M\_A\_OUT(M)\cup M\_A\_OUT'(P),M\_A'(M)=M\_A(M)\cup M\_A(P),$$

我们可按照与情况 a)~c)类似的讨论证明结论正确.

同理可证,$\rho(P)$也不会减小. □

定理 3. 在 MMG 中添加一条边,$CH(M_M)$不会减小.

定理 3 的证明与定理 1 类似,在此不再赘述.

定理 4. 设两个类的属性间的内聚度分别为 $CH_1(A_A)$和 $CH_2(A\_A)$,两个类简单合并后,内聚度为 CH_{12},则 $CH_{12}(A\_A)\leq \mathrm{Max}(CH_1(A\_A),CH_2(A\_A))$.

证明:两个类简单合并后,类中原有的各种依赖关系没有改变,也没有增加新的依赖关系.设类 C_1 和 C_2 分别有 n_1 和 n_2 个属性.

- 若 $n_1=n_2=0$,则 $CH_{12}(A\_A)=CH_1(A\_A)=CH_2(A\_A)=0$.
- 若 n_1 或 $n_2=1$,不妨设 $n_1=1$,则 $CH_1(A\_A)=1$.这是内聚度的最大值,因此 $CH_{12}\leq 1$,$CH_{12}(A\_A)\leq CH_1(A\_A)$.
- 若 $n_2,n_1>1$,令 $CH_1(A\_A)=\dfrac{\alpha_1}{n_1(n_1-1)}$,$CH_2(A\_A)=\dfrac{\alpha_2}{n_2(n_2-1)}$,

不妨设 $CH_1(A\_A)<CH_2(A\_A)$,即 $\alpha_1*n_2*(n_2-1)<\alpha_2*n_1*(n_1-1)$.

根据定义得 $CH_{12}(A\_A)=\dfrac{\alpha_1+\alpha_2}{(n_1+n_2)(n_1+n_2-1)}$.显然有,$CH_{12}(A\_A)<CH_2(A\_A)$.

因此,$CH_{12}(A\_A)\leq \mathrm{Max}(CH_1(A\_A),CH_2(A\_A))$. □

定理 5. 根据$\rho(M)$的定义,将两个类简单合并,$\rho(M)$保持不变.

定理 6. 设两个类的方法间的内聚度分别为 $CH_1(M_M)$和 $CH_2(M\_M)$,两个类简单合并后为内聚度 CH_{12},则 $CH_{12}(M\_M)\leq \mathrm{Max}(CH_1(M\_M),CH_2(M\_M))$.

定理 6 的证明过程与定理 4 类似.

5 子类的内聚度度量讨论

一般地,继承将增加类间的耦合度,降低内聚度.子类的内聚度度量可分 3 种情况讨论:

(1) 只考虑子类添加部分的内聚度,不考虑超类的元素.

这种方法的缺点是:只反映了增加部分的内聚度,无法真正反映类的本质.

(2) 将超类中除私有方法和属性的其他元素都加入到子类一起分析.

除了多态调用以外,超类中的方法不可能访问子类的属性和方法,因此,随着继承层次的加深,其内聚度度量结果会越来越小.

(3) 被子类引用的属性或方法与子类添加部分一起分析.

若子类方法直接或间接访问了超类的属性(或方法),则将这些属性(或方法)一起作为子类的属性(或方法)进行分析.若父类的方法通过多态调用可能访问子类的属性,则将这些方法与子类一起分析.

6 结 语

本文提出一种基于程序依赖性分析的类内聚度度量方法,它从属性与属性之间、属性与方法之间以及方法与方法之间 3 个方面对类进行全面分析.这 3 个方面既可以单独度量类的内聚度,也可以综合使用.我们还证明了本文提出的方法满足 Briand 提出的关于优良的内聚度度量准则的 4 个基本性质.一个优良的内聚度度量准则应该满足这些性质,但满足这些性质的内聚度度量准则不一定就是优良的内聚度度量准则.在将来的工作中,我们将通过实验来验证并改进我们的度量方法.

在度量类内聚度时,下面几个问题还需要进一步加以探讨:

(1) 在类继承层次中,超类可能通过多态方法访问子类的方法和属性.因此,在度量超类内聚度时,是否需要考虑对子类的访问值得进一步探讨.

(2) 在具体应用系统中,我们可以判定可能的多态调用.但是,对于一个可复用类而言,很难判断其多态调用情况.

(3) 在度量类内聚度时,如何分析一些特殊方法:如构造函数、访问方法等,因为这些方法一般只访问个别属性,若与其他方法一起分析,一般情况下会降低内聚度.

(4) 如何将域知识溶入内聚度度量中?

总之,若一个类可能应用于多个应用环境,则其内聚度度量主要考虑类本身的特性;若某个类是为特定应用开发的,则应考虑具体应用的情况.

References:

[1] Allen EB, Khoshgoftaar TM. Measuring coupling and cohesion: An information-theory approach. In: Proceedings of the 6th International Software Metrics Symposium. IEEE Computer Society, 1999. 119~127.

[2] Briand LC, Morasca S, Basili VR. Property-Based software engineering measurement. IEEE Transactions on Software Engineering, 1996,22(1):68~85.

[3] Briand LC, Daly JW, Wüst JK. A unified framework for Coupling measurement in object-oriented systems. Empirical Software Engineering, 1998,3(1):65~117.

[4] Briand LC, Morasca S, Basili VR. Defining and validating measures for object-based high-level design. IEEE Transactions on Software Engineering, 1999,25(5):722~743.

[5] Chae HS, Kwon YR. A cohesion measure for classes in object-oriented systems. In: Proceedings of the 5th International Software Metrics Symposium. IEEE Computer Society Press, 1998. 158~166.

[6] Chidamber SR, Kemerer CF. A metrics suite for object oriented design. IEEE Transactions on Software Engineering, 1994,20(6):476~493.

[7] Hitz M, Montazeri B. Measuring coupling and cohesion in object-oriented systems. In: Proceedings of the International Symposium on Applied Corporate Computing. 1995. 75~84.

[8] Xu BW, Zhou YM. Comments on 'A cohesion measure for object-oriented classes'. Software--Practice and Experience, 2001, 31(14):1381~1388.

[9] Chen ZQ, Zhou YM, Xu BW, Zhao JJ, Yang HJ. A novel approach to measuring class cohesion based on dependence analysis. In: IEEE International Conference on Software Maintenance (ICSM 2002). 2002. 377~383.

[10] Chen ZQ, Xu BW. Slicing object-oriented Java programs. ACM SIGPLAN Notices, 2001,36(4):33~40.

[11] Xu BW, Chen ZQ, Zhou XY. Slicing object-oriented Ada95 programs based on dependence analysis. Journal of Software, 2001, 12:208~213 (in Chinese with English abstract).

[12] Chae HS, Kwon YR, Bae DH. A cohesion measure for object-oriented classes. Software—Practice and Experience, 2000,30(12): 1405~1431.

附中文参考文献:

[11] 徐宝文,陈振强,周晓宇.基于依赖性分析的面向对象 Ada95 程序切片.软件学报,2001,12:208~213.