文章编号:1000-5862(2015)05-0459-04

# 一种面向对象的类级复杂性度量方法

秦怀斌,郭　理
(石河子大学信息科学与技术学院,新疆 石河子　832003)

**摘要**:从属性、操作、属性间、操作间、操作属性间等方面给出类内复杂性度量方法,再从节点的强度、簇系数、平均路径长度等方面给出类结构的复杂性度量方法,最后以具体系统的类图为例进行类结构的复杂性度量实证检验. 结果表明:该度量方法能较好地对面向对象类级复杂性进行度量.

**关键词**:面向对象;类;复杂性;度量
**中图分类号**:TP 311　　**文献标志码**:A　　**DOI**:10.16357/j.cnki.issn1000-5862.2015.05.04

## 0　引言

面向对象方法是目前的主流开发方法,由于继承、封装等特征,面向对象方法比其他软件开发方法更适合支持模块化、构件化软件开发及软件复用. 在面向对象软件开发中,面向对象软件度量的理论研究及实践应用是目前软件工程领域的研究热点. 类作为面向对象软件开发中至关重要的制品,类内部的复杂性及类之间的复杂性(类结构复杂性)不仅会影响到软件分析、设计、实现和测试的工作,而且对软件后续维护的影响也是深远的. 通过对类内部及类结构的复杂性进行度量,不仅可以对将要开发的软件的复杂性进行预警,而且可以有效控制软件系统的质量,并为后续的软件维护工作奠定良好的基础.

在面向对象复杂性度量研究中,Lorenz 于 1993 年提出了 11 个面向对象的度量准则及一些度量检验的规则;1994 年 S. R. Chidamber 和 C. F. Kemerer 提出了面向对象复杂性 C&K 度量集,同年,Brito 提出了 MOOD 度量,Lorenz 和 Kidd 提出了 LK 度量. 1993 年 J-Y Chen 和 J-F Liu 提出了 Chen&Liu 度量方法;Wei Li 基于 C&K 度量集,从面向对象继承类的个数、父类的个数、类外能访问的操作数等角度出发,于 1998 年提出了"另一个面向对象程序设计度量集"[1-4]. 此外,国内外还基于以上度量方法,进行了一定的扩展和完善,如 2004 年 Erik Arisholm 提出"动态"度量面向对象的耦合;Andrian Marcus 于 2005 年提出基于语义分析类之间"概念内聚度"的方法等[5-8].

以上所述方法,均只是给出了度量集,并没有给出详细的使用指导细节. 另外,C&K 只考虑了面向对象的类的操作和类的属性之间的关系,没有考虑类的属性之间、类的操作之间的关系;MOOD 度量集的度量粒度不够详细,且容易受到环境的影响;Wei Li 度量集缺乏对系统级的复杂性度量等. 针对以上问题,本文提出一种面向对象的类级复杂性度量方法,即从类的属性、类的操作角度,度量类内部的复杂性;然后从类之间关系的角度,对整个类结构的复杂性进行度量;最后,以新疆生产建设兵团开发的兵团空间信息系统中基于遥感(Remote Sensing,RS)和地理信息系统(Geographical Information System,GIS)的洪灾检测评估系统的类图为例,进行本研究中的类级复杂性度量实证检验. 验证结果表明,该度量方法可以实现对面向对象类级复杂性进行较好的度量[5,9-15].

## 1　类内部的复杂性度量

根据面向对象的类的概念,类包括属性和操作 2 个方面内容,并且属性之间、操作之间、操作与属性之间均可能会存在关系,如依赖、调用、使用等. 所以,类内部的复杂性度量需围绕属性、操作、属性间、操作间、操作属性间等方面的复杂性进行度量.

收稿日期:2015-03-02
基金项目:国家社会科学基金(14XXW004)和教育部人文社科研究新疆课题(11XJJAZH001)资助项目.
作者简介:秦怀斌(1980-),男,宁夏中卫人,副教授,主要从事软件工程、复杂网络的研究.

### 1.1 属性(Attribute)复杂性度量

**定义1** 属性的数量($A_1$):类包含的属性的个数,包括新定义的属性和继承的属性.

**定义2** 属性的复杂性($C_1$):$C_1 = \sum_{i=1}^{A_1} AC_i$,其中$AC_i$为每个属性的复杂性,$A_1$为属性的数量. 每个属性不一定具有相同的复杂性. 若每个属性均为简单的数据类型(如C语言中的int,float,char等),则认为每个属性的复杂性相同(如复杂性设置为单位1);若类中含有复杂的数据类型(如C语言中的struct,union等),则这个属性的复杂性要比简单的数据类型的复杂性高,在简单数据类型的基础上,可以设置这个复杂数据类型的复杂性(如根据复杂数据类型本身的层数、包含的成员数量及成员类型等设置).

### 1.2 操作(Method)复杂性度量

**定义3** 操作的数量($M_1$):类包含的操作的个数,包括新定义的操作和继承的操作.

**定义4** 操作的复杂性($C_2$):$C_2 = \sum_{i=1}^{M_1} MC_i$,其中$MC_i$表示每个操作的复杂性,$M_1$为操作的数量.

同属性一样,每个操作不一定具有相同的复杂性. 每个操作的复杂性可以根据操作实现的代码量、功能点数、算法复杂性、参数个数、参数类型等确定. 如以采用简单算法、包含1个简单数据类型参数、1个功能点、10行代码实现的操作为基准(如复杂性设置为单位1),然后去计算其他操作的复杂性.

### 1.3 属性间复杂性度量

如果类内2个属性之间有关联,若一个属性依赖于另外一个属性,则需要考虑属性间复杂性. 设定,若属性$i$依赖于属性$j$,则属性$i$与$j$间复杂性为1,否则,属性间复杂性为0. 并且,属性$i$依赖于属性$j$与属性$j$依赖于属性$i$是不同的.

**定义5** 属性间复杂性($C_3$):$C_3 = \sum_{i=1}^{A_1}\sum_{j=1}^{A_1} AAC_{ij}$,且$i \neq j$,其中$AAC_{ij}$表示属性$i$与属性$j$间的复杂性,$A_1$为属性的数量. 根据以上假设及分析可知,$AAC_{ij}$的取值为1或0.

### 1.4 操作间复杂性度量

若类内2个操作之间有关联,若一个操作调用另外一个操作,则需要考虑操作间复杂性. 设定,若操作$i$依赖于操作$j$,则操作$i$与$j$间复杂性为1,否则,操作间复杂性为0. 并且,操作$i$调用操作$j$与操作$j$调用操作$i$是不同的.

**定义6** 操作间复杂性($C_4$):$C_4 = \sum_{i=1}^{M_1}\sum_{j=1}^{M_1} MMC_{ij}$,且$i \neq j$,其中,$MMC_{ij}$表示操作$i$与操作$j$间的复杂性,$M_1$为操作的数量. 根据以上假设及分析可知,$MMC_{ij}$的取值为1或0.

### 1.5 操作属性间复杂性度量

**定义7** 操作属性间复杂性($C_5$):$C_5 = \sum_{i=1}^{M_1} MAC_i$,其中$MAC_i$表示第$i$个操作与属性间的复杂性,$M_1$为操作的数量. 若操作$i$使用或修改其中的1个属性,则操作$i$的操作属性间复杂性为1,若操作$i$使用或修改其中的2个属性,则其复杂性为2,…,以此类推. 根据假设及实际可知,$0 \leqslant MAC_i \leqslant A_1$.

上述为对单个类的复杂性进行度量,若某个类的复杂性过大,则需要按照软件工程思想对该类进行拆分,以降低该单个类的复杂性.

## 2 类结构的复杂性度量

在面向对象软件开发中,类图是描述类、接口以及他们之间的关系(结构)的工具. 类图贯穿于软件开发的全过程,类图的分析及设计也是软件开发的核心工作之一. 本节在类内部的复杂性度量基础上,研究类图(类结构)的复杂性的度量方法.

由于类之间的关系较多,如继承(inheritance)、聚合(aggregation)、关联(association)、消息(message)、依赖(dependency)等,这些关系存在较大差异,为了简化研究,也为了使研究方法能真正无障碍的应用到实际软件系统的开发中,本文将类图进行简化[4-6]:将类图转化为有向图(graph),用抽象的节点表示类,用抽象的有向边表示类间的关系,且每条有向边的复杂性设置为单位1(如果节点$a$与节点$b$有$n$种关系,则在节点$a$与节点$b$之间有$n$条有向边,且可能存在多条同向边;如果节点$a$调用节点$b$的$m$个操作,则节点$a$有$m$条边指向节点$b$). 即,一个类图抽象为由一个节点集$V$和边集$E$组成的图$G = (V,E)$,节点总数(类总数)$N = |V|$,边总数(关系总数)$D = |E|$.

### 2.1 节点的强度

用$W_{ij}$表示从节点$V_i$到节点$V_j$连接的边的复杂性值(1或0). 用$S_i$表示节点$V_i$的强度(strength). 节点$V_i$的强度是指与该节点直接相连接的边的总数. 节点的入强度指从其它节点直接指向此节点的

所有的边的总数,节点的出强度为从此节点直接指向其他的节点的所有边的总数.

**定义 8**　节点的入强度 $\overset{\vee}{S}_i = \sum_{j \in V(i)} W_{ji}$,其中 $W_{ji} = 1$ 或 0.

**定义 9**　节点的出强度 $\overset{\wedge}{S}_i = \sum_{j \in V(i)} W_{ij}$,其中 $W_{ij} = 1$ 或 0,$V(i)$ 代表节点 $V_i$ 的近邻集合.

根据定义 8 及定义 9 可知,节点的总强度为该节点的入强度和该节点的出强度之和.

**定义 10**　节点的总强度 $S_i = \overset{\vee}{S}_i + \overset{\wedge}{S}_i$.

通过度量可知,若节点的入强度越大,则说明与该节点对应的类的操作被较多类所调用,或较多类依赖或继承于该类; 若节点的出强度越大,则说明与该节点对应的类要过多的依赖于其他类、或调用其他类的操作(隐含了调用结果值的返回). 同理,与节点对应的类的总强度的高低,可以反映该类在整个软件系统中的重要程度. 节点的总强度越高,说明与该节点对应的类的影响面越大,反之,该类影响面较小.

### 2.2　簇系数

簇系数(clustering coefficient)用于衡量类结构的群组程度,即类结构中节点按照群组进行分布,群组内的节点之间的边的相连密集,而群组之间相关联的边较少. 类结构的簇系数反映了类结构中节点(类)之间的有向连通程度,或类结构中节点(类)的局部传递程度. 按照软件工程设计思想,将拟开发软件中的类按照包的机制划分为多个包(组),包内的类之间联系较为紧密,而包之间的连接较松散(即用简单的、较少的边连接).

若类结构中的一个节点类 $i$,有 $n_i$ 条边把它与其他节点类相连接,则与这 $n_i$ 条边相连接的 $n_i$ 个节点类之间,最多有 $n_i(n_i-1)/2$ 条边. 用这 $n_i$ 个节点类之间的实际存在边数 $E_i$ 与可能存在的边数 $n_i(n_i-1)/2$ 之比来定义为节点类 $i$ 的簇系数 $CC_i$,即 $CC_i = 2E_i/(n_i(n_i-1))$. 那么,整个类结构的簇系数就是类结构中的所有节点类的簇系数的平均值.

**定义 11**　类结构的簇系数 $CC = \frac{1}{N}\sum_{i=1}^{N} CC_i$,其中 $CC_i = 2E_i/(n_i(n_i-1))$,其中 $N$ 为类结构中类的总个数. 显然有 $CC \in [0,1]$. $CC = 0$ 表示类结构中所有的节点类均为孤立节点; $CC = 1$ 表示整个类结构中的节点是全连通的,并且任意 2 个节点类均是直接连接的.

### 2.3　平均路径长度

节点类的平均路径长度,指该节点类到其他的所有节点类最短路径的平均值. 类结构的平均路径长度,指该类结构中所有的节点类对之间最短路径的平均值,即任何 2 个节点类之间需经过的边数目的平均值.

**定义 12**　类结构的平均路径长度

$$L = \frac{1}{N(N-1)}\sum_{i \in V}\sum_{j \neq i \in V} d(i,j) \quad \text{或}$$

$$L = \frac{1}{N(N-1)/2}\sum_{i \geqslant j} d(i,j),$$

其中 $V$ 为节点类的集合,$N = |V|$ 为节点类的总个数,$d(i,j)$ 为节点 $i$ 到节点 $j$ 的最短距离. 类结构的平均路径长度说明了某节点类在类结构中的中心度. 节点类的平均路径长度越短,则该节点类与其他的节点类之间所经过的边数越少,说明该节点类在类结构中的中心度越高. 根据软件工程理论,软件系统的结构层数不能太深,若对类进行了层次划分,则类尽量依赖其紧邻下层的类,避免越级依赖,所以,类结构应该具有较小的平均路径长度.

## 3　实例验证

随着计算机技术及网络技术的发展,新疆生产建设兵团(以下简称“兵团”)在国家支持下开发了兵团空间信息系统,为兵团得屯垦戍边提供支服务. 该系统包括农业、水利、土地、生态环境和公共安全等 5 个子系统. 作为验证,本文选取了水利子系统中基于遥感(Remote Sensing,RS)和地理信息系统(Geographical Information System,GIS)的洪灾检测评估系统作为实例进行验证. 基于 RS 和 GIS 的洪灾检测评估系统采用面向对象方法开发,其类图如图 1 所示.

将图 1 的系统类图转化为如图 2 所示的类结构有向图. 转化后的类结构有向图用 $G = (V,E)$ 表示,其中,节点 $V$ 集合: $V = \{V_1, V_2, V_3, V_4, V_5, V_6, V_7, V_8, V_9, V_{10}, V_{11}\}$. 由于本类图度量中没有涉及到类之间的关系(如前所述,用抽象的有向边表示类间的关系,且每条有向边的复杂性设置为单位 1),故不对边集合 $E$ 进行描述,只关注类的连接情况.

通过分析,图 2 中 11 个节点类的总强度(入度与出度的和)分别为: $|V_1| = 2$, $|V_2| = 2$, $|V_3| = 2$, $|V_4| = 1$, $|V_5| = 4$, $|V_6| = 3$, $|V_7| = 2$, $|V_8| = 4$, $|V_9| = 2$, $|V_{10}| = 3$, $|V_{11}| = 1$. 同理,每个节点类的

入度和出度可分别计算得出. 通过分析可知,度为 4 的节点有 2 个,度为 1 的节点有 2 个,其余节点类的度均为 3 或 2. 节点的度分布符合软件工程设计提倡的正态分布.

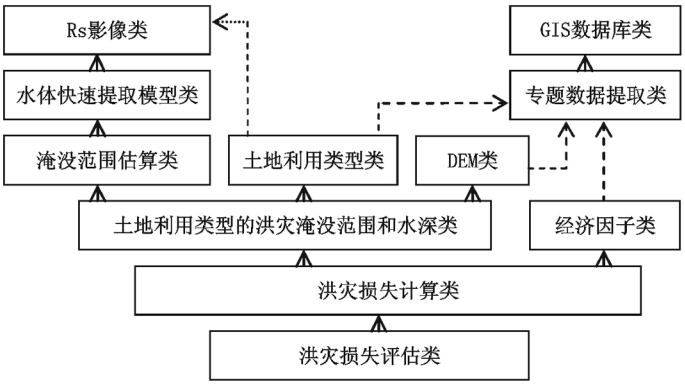


图 1 系统类图

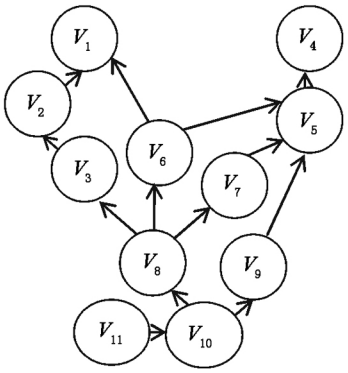


图 2 简化的系统类结构有向图

根据类结构的平均路径长度的计算公式 $L = \frac{1}{N(N-1)}\sum_{u\in V}\sum_{v\neq u\in V} d(u,v)$,考虑该系统类结构路径的有向图,通过计算,得出该系统类结构的平均路径长度值为 1.288. 同理,可计算该系统类结构的簇系数值和其他属性值. 分析得知,该子系统的类结构具有较合适的簇系数值和较短的平均路径长度值,系统的度分布比较合理,符合软件工程理论思想.

## 4 总结

类及类图作为面向对象软件开发中至关重要的制品,类的内部复杂性及类结构的复杂性直接影响到系统开发的各个阶段的工作. 本文提出一种面向对象的类级复杂性度量方法,通过对类内部及类之间(类结构)的复杂性进行度量,以此实现在系统开发的早期对系统的类及类结构复杂性进行度量和把握,为相关人员提供参考. 通过兵团空间信息系统软件系统的类图作为实例,验证了类结构的复杂性. 实例证明,该方法能比较好地度量所开发的系统类结构的复杂性,为相关分析及开发人员提供判断依据,有效提高所开发软件系统的可靠性及质量.

## 5 参考文献

[1] 曾一,柴艳欣,吴光金. 面向对象类的复杂性的度量方法 [J]. 计算机工程与应用,2010,46(12):64-67,71.

[2] 李大鹏,郭平,陈新宇. 一种集成类层次和系统层次的面向对象软件复杂性度量集 [J]. 计算机研究与发展,2010,47(增):237-242.

[3] 易彤. 面向对象设计中软件度量学:回顾与热点 [J]. 计算机应用研究,2011,28(2):427-434.

[4] 秦怀斌,郭理. 基于用例图的软件系统复杂网络特性度量 [J]. 微电子学与计算机,2012,29(7):72-75,80.

[5] 姜茸,杨明. 软件复杂性研究综述 [J]. 计算机系统应用,2014,23(9):1-5.

[6] 秦怀斌,李道亮,郭理. 基于复杂网络的软件体系结构复杂性度量方法 [J]. 微电子学与计算机,2013,30(2):5-8.

[7] 戴建国,梁斌,郭理. Hibernate 框架下信息系统数据初始化问题的研究 [J]. 电脑知识与技术,2010,6(7):1551-1553.

[8] 谢超超,杨柳. 一种优化的面向对象软件复杂性度量方法 [J]. 微型机与应用,2013,32(21):66-68,72.

[9] 张伟. 面向对象软件复杂性度量研究 [D]. 武汉:武汉理工大学,2006.

[10] 黄龙玲. 基于类图的面向对象软件复杂性度量方法的研究 [D]. 南昌:江西财经大学.

[11] 伦立军,丁雪梅,李英梅. 基于继承图的面向对象软件复杂性度量研究 [J]. 计算机工程与应用,2006,(27):97-99.

[12] 曾一,李娟,郭英君. 一种面向对象多态复杂性的度量方法 [J]. 计算机应用研究,2009,26(6):342-345.

[13] 李兵,马于涛,刘婧. 软件系统的复杂网络研究进展 [J]. 力学进展,2008,36(6):805-814.

[14] Wang Guiying,Zhou Jian,Xie Yang. Directed weighted network model based on BBV [J]. Computer Engineering,2010,36(12):141-143.

[15] Zhang Chengcai,Qi Xiaogang. Analysis of wireless sensor network characteristics measurement based of complex network theory [J]. Computer Science,2011,37(2):44-46.

(下转第 468 页)

# The Algorithm of Dynamic Chain Address for Mining Association Rules Based on DHP

WU Heng, WU Genxiu*, MAO Linchuan, HUANG Mei
(School of Mathematics and Information Science, Jiangxi Normal University, Nanchang Jiangxi 330027, China)

**Abstract:** DLDHP algorithm solves DHP algorithm's problem that can't separate count for each candidate itemsets by using dynamic chain address to create Hash table. Increasing the table space and also deleting the nodes make Hash table won't occupy large space. Finally, frequent itemsets directly by the Hash table without scanning the database once again has been got. Examples show that DLDHP algorithm is effective and feasible.
**Key words:** DHP algorithm; DLDHP algorithm; Hash table; dynamic chain address

(责任编辑: 冉小晓)

(上接第 462 页)

# The Metrics Method of Object-Oriented Class Complexity

QIN Huaibin, GUO Li
(College of Information Science and Technology, Shihezi University, Shihezi Xinjiang 832003, China)

**Abstract:** The metrics methods of within class from attribute, method, between attributes, between methods, between methods and attributes, the metrics methods of class structure from node strength, clustering coefficient, average path length, finally with the specific system class diagram as an example to measure empirically complexity class structure have been given. The result shows that the metrics method can measure better object-oriented class complexity.
**Key words:** object-oriented; class; complexity; metrics

(责任编辑: 冉小晓)