# 面向对象系统的类之间依赖关系度量研究

欧 阳[1] 胡顺仁[1,2] 汪治华[1,2]

(重庆工学院网络中心 重庆400050)[1] (重庆大学光电工程学院 重庆400044)[2]

**摘 要** 类之间的依赖关系,对于面向对象系统分析、设计和测试都有重要的意义。本文首先对类之间的依赖关系进行了定义和说明,并细分其为数据依赖和方法依赖,在此基础上,提出依赖度和被依赖度两种度量方法,并进行了严格的语义分析和说明。

**关键词** 面向对象系统,依赖关系,数据依赖,方法依赖,依赖度,被依赖度

## Measurement Research of Dependency Relations among Classes Based on Object-Oriented System

OU Yang[1] HU Shun-Ren[1,2] WANG Zhi-Hua[1,2]

(Chongqing Institute of Technology, Chongqing 400050)[1]

(College Of Optoelectronic Engineering, Chongqing University, Chongqing 400044)[2]

**Abstract** The dependency relations among classes are important for object-oriented analysis, design and test. In this paper, the define of Class Dependency is first introduced, and fractionated into data dependency and method dependency, then, this paper describes the two methods: depending metrics and depended metrics.

**Keywords** OOS, Dependency relation, Data dependency, Method dependency, Depending metrics, Depended metrics

现实世界的问题是由客观实体和实体之间的联系构成的,对象就是客观实体的抽象,而对象之间的联系是复杂、多变的。对于规模较小的软件系统来说,搞清对象间的联系也许并不难,但是面向对象系统(OOS)开发往往都是规模较大的实体空间,对象数目众多、对象之间的关系错综复杂,在面向对象分析和设计(OOA&D)中很难作出完整、正确的判断,尤其是对象之间的依赖关系[1~3]。

类是面向对象系统组织结构的核心,是用来理解和描述众多对象的个别概念。类之间依赖关系即是对象之间的依赖关系的抽象。国内外许多OOS研究只是对类之间的依赖关系进行定性分析,缺乏定量研究。所以,对类之间的依赖关系的度量,在今天显得尤为重要。

## 1 类之间依赖关系的定义

类在面向对象方法学中的形式定义[4]是:类:=(ID, INH, DD, OI, ITF}。其中:ID为类的标识或类名;INH为类的继承性描述;DD为数据结构描述;OI为操作集合描述;ITF为对外接口。

由类的形式化定义可以看出,类是一种具有相同属性和相同操作的对象的集合。类是所有具有共同行为特征和信息结构的对象的集合。它代表一种抽象,是代表对象的本质的、主要的、可观察的行为。它给出了属于该类的全部对象的抽象定义,包括类的属性、操作和其他性质。每个类不是孤立、分割的,而是相互联系、制约的。类之间的依赖性也正是这种关系的体现。

在面向对象系统OOS中,S为OOS中的所有类的集合,记为:$S=\{C_1, C_2, \cdots, C_n\}$,其中,n为集合S中类的个数。我们定义类之间的依赖关系如下:

**定义1** 依赖关系D为定义在集合S上的二元关系,设存在两个任意的类$C_i$和$C_j$,且$C_i \in S, C_j \in S$,如果类$C_i$被修改、删除或移动后,则类$C_j$也会出现相应的变化,称$C_j$依赖于类$C_i$,记为:

$$D=\{\langle C_j, C_i\rangle \mid C_i \in S, C_j \in S\}$$

上述定义基本上描述了依赖关系的特点,但是这种描述是粗略和不完善的,没有能揭示出类具体是如何被修改、删除或移动的,所以我们需要重新定义依赖关系。

由于每个类都有两个基本特征:属性和操作。可以将类看作为属性集合和操作集合的集合,类$C_i$和$C_j$记为:

$$C_i=\{A_i, M_i\}, C_j=\{A_j, M_j\}$$

其中,$A_i=\{a_{i1}, a_{i2}\cdots, a_{is}\}$,$M_i=\{m_{i1}, m_{i2}\cdots, m_{iu}\}$。

$A_i$为类$C_i$中所有属性的集合,$M_i$为类$C_i$中所有操作的集合。s, u为类$C_i$属性和操作的数目。

$$A_j=\{a_{j1}, a_{j2}\cdots, a_{jt}\}, M_j=\{m_{j1}, m_{j2}\cdots, m_{jv}\}$$

$A_j$为类$C_j$中所有属性的集合,$M_j$为类$C_j$中所有操作的集合。t, v为类$C_j$属性和方法的数目。

那么,类$C_i$和$C_j$之间所有的各种关系都是$((A_i \cup M_i)\times(A_j \cup M_j)) \cup ((A_j \cup M_j)\times(A_i \cup M_i))$的子集($\times$为笛卡儿乘积),本文研究的类之间的依赖关系D也是其中的一个子集,定义如下:

**定义2** 设类$C_i$和$C_j$之间的依赖关系D为定义在集合$(A_i \cup M_i), (A_j \cup M_j)$上的二元关系,记为:

$$D=\{\langle x, y\rangle \mid (x \in (A_i \cup M_i) \wedge y \in (A_j \cup M_j)) \vee (x \in (A_j \cup M_j) \wedge y \in (A_i \cup M_i))\}$$

其中,序偶$\langle x, y\rangle$表示元素x依赖于元素y,即元素y的变化会引起元素x的变化。

欧 阳 副教授,系统工程专业硕士,主要研究方向为网络安全、软件工程和系统集成。胡顺仁 讲师,计算机专业硕士,主要研究方向为网络安全、软件工程,光电技术。

## 2 依赖关系的两种基本细分类型

如前面所述，类具有属性和操作这两个基本的特征，类之间的依赖关系也是这两个基本性质外部表现。类之间依赖性的产生，正是类的属性和操作之间的相互依赖所表现。我们将为属性所引起的依赖称为数据依赖，记为 $R_{DD}$；另一个为操作所引起的依赖称为方法依赖，记为 $R_{MD}$。这两种依赖类型实际上是类之间依赖关系的细分，其定义和描述如下：

### 2.1 数据依赖[5]

**定义3** 设存在两个属性 $a_{ie}$ 和 $a_{jf}$，且 $a_{ie}\in A_i$，$a_{jf}\in A_j$，若在类 $C_j$ 中存在属性 $a_{jf}$ 对类 $C_i$ 的属性 $a_{ie}$ 的引用或类 $C_i$ 中属性 $a_{ie}$ 通过参数形式传递给类 $C_j$ 的属性值 $a_{jf}$ 的情况，则称属性 $a_{jf}$ 数据依赖于数据 $a_{ie}$，记为 $a_{jf}\ R_{DD}\ a_{ie}$。

**定义4** 设存在一个属性 $a_{ie}$，$a_{ie}\in A_i$，一个操作 $m_{jv}\in M_j$，若在类 $C_j$ 中存在操作 $m_{jv}$ 对类 $C_i$ 中的属性 $a_{ie}$ 调用的情况，则称操作 $m_{jv}$ 数据依赖于属性 $a_{ie}$，记为 $m_{jv}\ R_{DD}\ a_{ie}$。

综上定义3，4可得：

**定义5** 设存在属性 $a_{ie}$，$a_{ie}\in A_i$，若类 $C_i$ 中的属性 $a_{ie}$ 的修改、删除、移动会引起类 $C_j$ 中属性和操作的相应变化，则称类 $C_j$ 数据依赖于类 $C_i$，记为 $C_j$ RDD $C_i$。

### 2.2 方法依赖

面向对象系统中存在各种各样的操作，由于类的继承性、多态性、动态联编、消息调用等因素，使得方法依赖变得要比数据依赖复杂。所以，我们研究方法依赖只能局限在静态上描述。

**定义6** 设两操作 $m_{iu}$、$m_{jv}$，且 $m_{iu}\in M_i$，$m_{jv}\in M_j$，如果存在操作 $m_{jv}$ 调用操作 $m_{iu}$ 或操作 $m_{jv}$ 接受操作 $m_{iu}$ 的消息激励的情况，则称 $m_{jv}$ 方法依赖于 $m_{iu}$，记为 $m_{jv}\ R_{MD}\ m_{iu}$。

**定义7** 设两操作 $m_{iu}$、$m_{jv}$，且 $m_{iu}\in M_i$，$m_{jv}\in M_j$，若类 $C_i$ 是类 $C_j$ 的子类，并且存在操作 $m_{jv}$ 为类 $C_j$ 中的虚拟函数，而操作 $m_{iu}$ 为操作 $m_{jv}$ 的同名函数，则称 $m_{iu}$ 方法依赖于 $m_{jv}$，记为 $m_{iu}\ R_{MD}\ m_{jv}$。

综上定义6，7可得；

**定义8** 设两操作 $m_{iu}$、$m_{jv}$，且 $m_{iu}\in M_i$，$m_{jv}\in M_j$，若类 $C_i$ 中的操作 $m_{jv}$ 的修改、删除、移动会引起类 $C_j$ 中操作的相应变化，则称操作 $C_j$ 方法依赖于操作 $C_i$，记为 $C_j\ R_{MD}\ C_i$。

## 3 依赖关系度量的理论分析

在度量类之间的依赖关系之前，我们首先给出依赖度和被依赖度的定义：

**定义9** 面向对象系统中，依赖度为一个类对另一个类依赖的程度，被依赖度为一个类被另一个类依赖的程度。

### 3.1 两个类之间的度量

1. *绝对度量* 我们可以简单定义如下：

**定义10** 类 $C_i$ 对类 $C_j$ 的绝对依赖度 $ad_{C_i}$ 是类 $C_i$ 指向类 $C_j$ 的数据依赖和方法依赖关系数目。

**定义11** 类 $C_i$ 对类 $C_j$ 的绝对被依赖度 $ab_{C_i}$ 是类 $C_j$ 指向类 $C_i$ 的数据依赖和方法依赖关系数目。

根据上述定义，我们得到如下性质：

**定理1** 类 $C_i$ 对类 $C_j$ 的绝对依赖度 $ad_{C_i}=|D\cap((A_i\cup M_i)\times(A_j\cup M_j))|$。

其有效取值范围为：$0\leqslant ad_{C_i}\leqslant(s+u)*(t+v)$。

**定理2** 类 $C_i$ 对类 $C_j$ 的绝对被依赖度 $ab_{C_i}=|D|-ad_{C_i}$。

其有效取值范围为：$0\leqslant ab_{C_i}\leqslant(s+u)*(t+v)$。

**定理3** $ad_{C_j}=ab_{C_i}$，$ab_{C_j}=ad_{C_i}$。

定理3表明这样一个性质：对于存在依赖关系的两个类，一个类的绝对依赖度即为另一个类的绝对被依赖度。这个性质我们从类的有向依赖图中很容易得到推论。

性能分析：当绝对依赖度 $ad_{C_i}=0$ 时，表示类 $C_i$ 不依赖于类 $C_j$，称这种关系为零关系，即类 $C_i$ 零依赖于类 $C_j$；当 $ad_{C_i}+ab_{C_i}=(s+u)*(t+v)$ 时，表示类 $C_i$ 的所有属性、方法和类 $C_j$ 的所有属性、方法都存在依赖关系，我们称这种依赖关系为满关系，又称类 $C_i$ 和类 $C_j$ 互为满依赖；当绝对依赖度 $ad_{C_i}=0$ 且绝对被依赖度 $ab_{C_i}=0$ 时，表明类 $C_i$ 和类 $C_j$ 之间不存在依赖关系，称这种关系为独立关系。在实际情况中，独立关系是常常存在，满关系一般是不可能存在的。

绝对度量基本上反映了类之间的依赖和被依赖的程度，在类的依赖关系不是很复杂的情况下可以使用。它最大的优点就是应用简单、方便。但是，其缺点也是明显的，没有考虑类的规模对它的影响，由于面向对象系统所应用的问题空间都是规模较大、关系复杂的领域，这种简单的度量在大多数场合往往不能满足用户的需要。

2. *相对度量* 根据前面的数学分析，定义如下：

**定义12** 类 $C_i$ 对类 $C_j$ 的相对依赖度 $rd_{C_i}$ 为类 $C_i$ 指向类 $C_j$ 的数据依赖和方法依赖关系数目与类 $C_i$ 和 $C_j$ 之间总关系数目的比值。

**定义13** 类 $C_i$ 对类 $C_j$ 的相对被依赖度 $rb_{C_i}$ 为类 $C_j$ 指向类 $C_i$ 的数据依赖和方法依赖关系数目与类 $C_i$ 和 $C_j$ 之间总关系数目的比值。

根据上述定义，我们可得到如下性质：

**定理4** 类 $C_i$ 对类 $C_j$ 的相对依赖度 $rd_{C_i}$，记为

$$rd_{C_i}=\frac{|DI((A_i\,YM_i)\times(A_j\,YM_j))|}{|(A_i\,YM_i)\times(A_j\,YM_j)|}$$

其有效取值范围为：$0\leqslant rd_{C_i}\leqslant 1$

**定理5** 类 $C_i$ 对类 $C_j$ 的相对被依赖度 $rb_{C_i}$，记为

$$rb_{C_i}=\frac{|DI((A_j\,YM_j)\times(A_i\,YM_i))|}{|(A_i\,YM_i)\times(A_j\,YM_j)|}$$

根据上述两个定义，很显然我们得到如下结论：

**定理6** $rd_{C_j}=rb_{C_i}$，$rb_{C_j}=rd_{C_i}$

其有效取值范围为：$0\leqslant rb_{C_i}\leqslant 1$

性能分析：当相对依赖度 $rd_{C_i}=0$ 时，表示类 $C_i$ 不依赖于类 $C_j$，这种关系为零关系，即类 $C_i$ 零依赖类 $C_j$；当 $rd_{C_i}+rb_{C_i}=1$ 时，表示类 $C_i$ 的所有属性、方法和类 $C_j$ 的所有属性、方法都存在依赖关系，这时的关系为满依赖；当 $rd_{C_i}=0$ 且 $rb_{C_i}=0$ 时，表明类 $C_i$ 和类 $C_j$ 之间不存在依赖关系，这时关系为独立关系。

相对度量应该说是比较科学、客观地反映了两个类之间相互依赖的程度，它不仅考虑依赖关系的数目，而且还用笛卡儿乘积来度量两个类之间的总关系，包含了两个类之间关系的规模大小。

3. *依赖性度量的细分* 在前面依赖性分类中，我们将类之间的依赖关系细分为数据依赖和方法依赖。这两种类型的依赖可以单独存在，也可以同时存在两个类或同一个类之间。两个类或同一个类之间也可以存在一个或多个数据依赖、方法依赖。我们在前述基础上，可以对两个类之间的依赖关系度量进行细分，有利于在 OOD 中更全面、详细地把握类之间的依赖特性。

**定义14**(数据依赖度) 描述一个类 $C_i$ 对另一个类 $C_j$ 数据依赖的程度，即类 $C_i$ 指向类 $C_j$ 的数据依赖关系数目与类 $C_i$ 和 $C_j$ 之间总关系数目的比值，记为 $dd_{C_i}$，则

$$dd_{C_i}=\frac{|DI(A_i\times(A_jYM_j))|}{|(A_iYM_i)\times(A_jYM_j)|}$$

其有效取值范围为：$0\leqslant dd_{C_i}\leqslant 1$

**定义15**(方法依赖度) 描述一个类 $C_i$ 对另一个类 $C_j$ 方法依赖的程度，即类 $C_i$ 指向类 $C_j$ 的方法依赖关系数目与类 $C_i$ 和 $C_j$ 之间总关系数目的比值记为 $md_{C_i}$，则

$$md_{C_i}=\frac{|DI(M_i\times(A_jYM_j))|}{|(A_iYM_i)\times(A_jYM_j)|}$$

其有效取值范围为：$0\leqslant md_{C_i}\leqslant 1$

数据依赖度、方法依赖度是对类之间的依赖程度的进一步细化描述，它比依赖度、被依赖度对类依赖程度的描述更具体。很显然，我们从其定义上可以得到如下性质：

**定理7** $dd_{C_i}+md_{C_i}=d_{C_i}$

性能分析：数据依赖度 $dd_{C_i}=0$ 时，表示类 $C_i$ 的属性不依赖类 $C_j$ 的属性和方法；同样，方法依赖度 $md_{C_i}=0$，表示类 $C_i$ 的方法不依赖类 $C_j$ 的属性和方法；当 $dd_{C_i}=0$ 且 $md_{C_i}=0$ 时，得 $d_{C_i}=0$，这时的关系为零关系；当 $dd_{C_i}+md_{C_i}=1$ 时，得 $d_{C_i}=1$，这时的关系为满关系。

### 3.2 依赖性度量的推广

上述依赖性度量的研究分析都是在二元关系的基础上开展的，在实际的面向对象系统研究中，类的依赖关系更多的体现在多个类之间，我们在前述二元关系研究的基础上，将依赖度和被依赖度等概念引申到类的多元关系。当然，面向对象系统中类的数目众多，不可能也没有必要研究系统中一个类与系统中所有类之间的依赖关系，我们常常将其中一些具有一定逻辑联系关系相连的类群作为研究对象。本文研究是在 UML 基础之上，将一张类图上的所有类作为一个类群来分析。类群定义如下：

**定义16** 类群 CM 为定义在类图 CG 上所有类的集合，记为 MG=｛x|x∈类图 CG 上的类｝。

这样，我们在类群 MG 的基础上，研究类依赖关系的度量。

设 $C_i=\{A_i,M_i\}$，其中，$C_i\in MG$，$A_i$ 为类 $C_i$ 中属性的集合，$M_i$ 为类 $C_i$ 中方法的集合。$n=|MG|$，n 为类群 MG 中类的数目，$i=1,2\cdots n$。

MD 为类群 MG 上所有类之间的依赖关系。

#### 1. 绝对度量

**定义17** 类 $C_i$ 对类群 MG 的绝对依赖度 $ad_{C_i}$ 是由类 $C_i$ 指向 MG-｛$C_i$｝中的类的数据依赖和方法依赖关系数目。

**定义18** 类 $C_i$ 对类群 MG 的绝对被依赖度 $ab_{C_i}$ 是 MG-｛$C_i$｝中的类指向类 $C_i$ 的数据依赖和方法依赖关系数目。同样，根据上述定义，我们得到如下性质：

**定理8** 类 $C_i$ 对类群 MG 的绝对依赖度：

$$ad_{C_i}=|MD\cap((A_i\cup M_i)\times(A_1\cup M_1))\cap((A_i\cup M_i)\times(A_2\cup M_2))\cap\cdots\cap((A_i\cup M_i)\times(A_{i-1}\cup M_{i-1}))\cap((A_i\cup M_i)\times(A_{i+1}\cup M_{i+1}))\cap\cdots\cap((A_i\cup M_i)\times(A_n\cup M_n))|=|MD\cap\overset{n}{\underset{j=1,j\neq i}{Y}}((A_iYM_i)\times(A_jYM_j))|$$

**定理9** 类 $C_i$ 对类群 MG 的绝对被依赖度：

$$ab_{C_i}=|MD\cap((A_1\cup M_1)\times(A_i\cup M_i))\cap((A_2\cup M_2)\times(A_i\cup M_i))\cap\cdots\cap((A_{i-1}\cup M_{i-1})\times(A_i\cup M_i))\cap((A_{i+1}\cup M_{i+1})\times(A_i\cup M_i))\cap\cdots\cap((A_n\cup M_n)\times(A_i\cup M_i))|=|MD\cap\overset{n}{\underset{j=1,j\neq i}{Y}}((A_iYM_j)\times(A_iYM_i))|$$

上述定理是很容易从定义中推导出来，证明过程省略。

#### 2. 相对度量

**定义19** 类 $C_i$ 对类群 MG 的相对依赖度 $rd_{C_i}$ 为类 C 指向 MG-｛$C_i$｝中类数据依赖和方法依赖关系数目与类 $C_i$ 和 MG-｛$C_i$｝总关系数目的比值。

**定义20** 类 $C_i$ 对类群 MG 的相对被依赖度 $rb_{C_i}$ 为 MG-｛$C_i$｝中的类指向类 $C_i$ 的数据依赖和方法依赖关系数目与类 $C_i$ 和 MG-｛$C_i$｝总关系数目的比值。

同样，根据上述定义，我们可以得到如下性质：

**定理10** 类 $C_i$ 对类群 MG 的相对依赖度：

$$rd_{C_i}=ad_{C_i}/\overset{n}{\underset{j=1,j\neq i}{Y}}((A_iYM_i)\times(A_jYM_j))$$

**定理11** 类 $C_i$ 对类群 MG 的相对被依赖度：

$$rd_{C_i}=ab_{C_i}/\overset{n}{\underset{j=1,j\neq i}{Y}}((A_iYM_i)\times(A_jYM_j))$$

**结束语** 在面向对象系统分析中，依赖度反映的是类的稳定性，一个类对其他的类的依赖程度越大，这个类就越不稳定，相关类就需要重新认真规划设计，尽量降低类的依赖度；被依赖度反映的是一个类的重要性，一个类被其他类依赖的程度越深，则这个类设计的优劣对于系统的影响就越大，在设计和测试过程中，该类就应该作为重点对象。但是，类的依赖度和被依赖度受多方面因素制约，不能简单地说高好，还是低好，特别是针对一个具体的类，要寻求一个合适的依赖度来满足不同的需要。

## 参 考 文 献


1 Priestley M. Practical Object-Oriented Design With UML. McGraw-Hill Companies, Inc, 2000. 111～152

2 Booch G, Rumbaugh J, Jacobson I. Unified Modeling Language User Guide(in Chinese). Beijing: China Machine press, 2001

3 Rumbaugh J, Jacobson I, Booch G. Unified Modeling Language Reference Manual (in Chinese). Beijing: China Machine Press, 2001

4 Zhou zhi-ying. Modern Software Project( in). Scientific Press (in Chinese), 2000, 12(3): 72～75

5 Fang fei, et al. An Approach to Object-Oriented Software Regression(in Chinese) Testing. Journal of Software, 2001, 12(3)

6 Jones C. Applied Software Measurement. Mcgraw-hill, 1986. 36～126

7 Fenton N E. Software Metrics, A Rigorous Approach. New York: Chapman & Hall, 1991. 45～78

8 Chidamber R, Kemerer F. A metrics suite for object-oriented design. IEEE Trans on Software Engineering, 1994, 20: 45～66

9 Zhuo Xiaoling, et al. Discrete Mathematics (in Chinese). Shanghai Science and Technology Press, 1982

10 Zhao J. Dyamic Slicing Object-Oriented Programs: [Technical Report. SE-9811]. Information Processing Society of Japan, 1998. 17-23. http://www.fit.ac.jp/～zhao.

11 Jacobson I, booch G, Rumbaugh J. the Unified Software Development Process. Addison Wesley Longman, Inc, 1999. 26～80

12 周之英. 现代软件工程(中). 科学出版社, 2000. 358～428

13 方菲, 等. 面向对象软件回归测试技术研究. 软件学报, 2001, 12(3)

14 左孝凌, 等. 离散数学. 上海科学技术文献出版社, 1982