面向对象 软件度量学 软件设计

87-89

计算机科学1998 Vol. 25 №. 1

面向对象设计中软件度量学的理论和方法*)

The Software Metrics Theory and Method in the Object-Oriented Design

邢大红 刘宗田

(合肥工业大学微机所 合肥230009)

TP311

摘 要 This article explains Chidamber and Kemerer's software metrics theory and method in the Object-Oriented design and discusses its application and shortage.

关键词 Object-Oriented, Software metrics, Class, Inheritance tree

随着面向对象技术的兴起,软件设计方法引入了面向对象设计技术(OOD)。数据抽象、封装、继承、多态性、信息隐藏、重用机制等新概念的出现,必然需要新的面向对象设计的软件度量学理论和方法与之适应。Chidamber 和 Kemerer 从91年就开始研究面向对象设计中的软件度量学[1][3],94年发表的《A Metrics Sutie for Object Oriented Design》[2]奠定了面向对象设计中的软件度量学的基础。本文阐述了面向对象设计中的软件度量学的理论基础,定义了有关 OO 度量学术语,阐述了面向对象设计中的软件度量方法和意义,讨论了 Chidamber 和 Kemerer 的面向对象设计中的软件度量学的理论和方法存在的问题。

一、OOD 中的软件度量学的理论基础

Roberts 把关系系统定义成一个顺序元组,这个顺序元组包括一组元素、一组关系和一组操作[8]。面向对象设计(OOD)也可以这样使之概念化。在 OOD 中关系系统 D 形式化描述如下:

$$D\equiv(A,R_1\cdots R_n,O_1\cdots O_m)$$

其中:A 是一组对象元素;$R_1\cdots R_n$ 是 A 中元素的经验关系(例如大于、小于等);$O_1\cdots O_m$ 是 A 中元素二进制操作(例如合并)。

下面以度量复杂性为例,我们来理解一组对象上的经验关系。一般,设计者对不同的对象元素有直觉的概念,认为这个对象元素比另一个复杂或复杂性相等。一个设计者可以凭直觉判断一个有很多方法的类总是更复杂些。这种直觉概念可以原原本本地被引用作为评价度量。Fenton[4]认为,一个观点不仅要表达出直觉理解的特点,还要从逻辑出发为度量定义,从而保证经验关系和观点一致。

二进制关系≥定义在集合 P 上(P 是所有可能设计的集合之一)。对 p,p',p''∈P 有以下两条公理:

1. p≥p'或 p'≥p(完备性:p 比 p'复杂或 p'比 p 复杂)。

2. p≥p',p'≥p''⇒p≥p''(传递性:若 p 比 p'复杂,p'比 p''复杂,则 p 比 p''复杂)。

为了能度量对象设计,上面定义的经验关系系统需要映射成形式关系系统。形式关系系统 F 定义如下:

$$F\equiv(C,S_1\cdots S_n,B_1\cdots B_m)$$

其中:C 是一组元素(例如实数);$S_1\cdots S_n$ 是 C 中元素上的形式关系(例如>,<,=);$B_1\cdots B_m$ 是集合上元素的二进制操作(例如+,-,×,÷)。

把关系系统 D 映射到形式关系系统 F 需要一个度量 μ 来实现,这个 μ 要保留构成经验关系的内涵而不能改变它。

二、OOD 中的软件度量学术语定义

不少研究者把 Bunge 的"基础哲学论"中定义的对象概念引用到 OO 领域。虽然 Bunge 没有为 OO 概念建立经验关系,但是 Bunge 对世界的表示和定义完全达到了 OO 方法的目的。一个实体和它的属性共同构成一个对象。因此,一个对象并不简单地是一组方法,而是包括设计者分配给此对象的方法和实例。一个对象可以用下列方式表示:

*)本文受机械工业技术发展基金资助。邢大红 硕士生,主要从事面向对象方法和软件度量学研究。刘宗田 博士生导师,研究员,主要从事软件工程、软件逆向工程和面向对象方法研究。

$X=\langle x,p(x)\rangle$

其中：x 是实体；p(x)是实体的属性的有限集合。

x 可以认为代表一个系统中对象的名字。在 OO 术语中，实例变量和它的方法就是对象的属性。下面定义6个 OO 术语。

1. **耦合**(Coupling)。设 $x=\langle x,p(x)\rangle$，$Y=\langle y,p(y)\rangle$是两个对象，有：

$p(x)=\{M_x\}\cup\{I_x\}$

$p(y)=\{M_y\}\cup\{I_y\}$

这里$\{M_i\}$是对象 i 的一组方法；$\{I_i\}$是对象 i 的一组实例变量。

任何由$\{M_x\}$作用于$\{M_y\}$或$\{I_y\}$就构成耦合关系，同样地，$\{M_y\}$作用于$\{M_x\}$或$\{I_x\}$也构成耦合关系。

2. **内聚**(Cohesion)。Bunge 定义的相似度 $\sigma()$是由两个事物的一组属性的共同部分组成：

$\sigma(x,y)=p(x)\cap p(y)$

Bunge 的相似度原理可以引用来定义对象中方法的相似度：

$$\sigma(M_1,M_2)=\{I_1\}\cap\{I_2\}$$

其中 $\sigma(M_1,M_2)$是方法 M_1和 M_2的相似度；I_i 是被方法 M_i 使用的实例变量的集合。

例：设$\{I_1\}=\{a,b,c,d,e\}$，$\{I_2\}=\{a,b,e\}$

则$\{I_1\}\cap\{I_2\}$不空，$\sigma(M_1,M_2)=\{a,b,e\}$

方法相似度既描述了传统的软件工程的内聚概念，也描述了 OO 的封闭性。它可以被看作为面向对象中类的内聚性的一个重要方面。假如一个对象类有不同的方法在相同的实例变量集合上完成不同的操作，则这个类是内聚的。这个内聚性的观点是集中于封闭在对象中的数据以及封闭在对象中的方法如何与数据相互作用方面上的。

3. **对象的复杂性**。Bunge 定义一个个体的复杂性是“成员的数目”，即一个复杂的个体有很多的属性。同样，一个类的复杂性可以被定义为它的属性集合的基数。

$X=\langle x,p(x)\rangle$的复杂性$=|p(x)|$

其中：$|p(x)|$是 p(x)的基数

4. **属性的作用域**。设计者对应用领域进行抽象，以层次形式安排类。这种继承层次可以是一个直接的非循环的图，可以被描述成一个树结构。类可以作为树中的结点、叶结点或根结点。在任何应用中，类的层次可以有很多选择，设计者通过这种安排可以限制或扩大属性的作用域。为此，这里定义两个与继承层次有关的术语：继承深度 DOI(Depth of Inheritance)和直接孩子数 NOC(Nubmer Of Children)。

DOI＝类在继承树中的层次

NOC＝类的直接孩子数目

DOI 和 NOC 描述了属性的影响程度。通过 DOI 可以揭示这个类受它的祖先类的影响程度。通过 NOC 可以揭示这个属性对它的子孙类的影响。

5. **作为通讯度量的方法**。在面向对象方法中，对象的通讯主要通过消息传递。方法可以被看作为对可能消息的反应。

对象类的响应集合＝{所有可以被调用对消息的响应的方法的集合}

注意这个集合包括了这个类以外的方法，因为这个类的方法可以调用别的类的方法。这个响应集合是有限的，因为设计中类的个数有限，每个类的属性也是有限的。

6. **对象类的合并**。类的设计过程可以通过现有的类来设计新类，这就是子类，它反应了继承关系。还可以把类分裂成更小的类，这样加强了类的内聚性，减少了类的复杂性。还可以把现有的类合并成一个类，两个类的合并产生的新类属性是成员类的属性的并集。

设 $X=\langle x,p(x)\rangle$，$Y=\langle y,p(y)\rangle$是两个对象类。则

X＋Y 被定义为$\langle z,p(z)\rangle$

这里 Z 是代表 X＋Y 的名字，$p(z)=p(x)\cup p(y)$

三、OOD 中度量方法及应用意义

Chidamber 和 Kemerer 定义了6个度量指标。

Metric 1：每个类的加权方法数 WMC (Weighted Methods per Class)

定义1 设类 C_i 有方法 $M_1\cdots M_n$，$C_1\cdots C_n$ 分别为 $M_1\cdots M_n$ 的复杂度，则：

$$WMC=\sum_{i=1}^{n}C_i$$

假如所有方法复杂度定义为1，则 WMC＝n，即类中的方法数之和。这里具体方法中的复杂度是用传统的软件设计中复杂度来计算的。

WMC 的应用意义：①方法数目和方法的复杂度可以预测开发和维护一个类需要的时间和难度。②一个类的方法数目越大，则对孩子类的可能影响越大。③有很多方法的类可能是具体的某个应用，这就限制了重用。

Metric 2：继承树的深度 DOI(Depth Of Inheritance)

定义2 DOI 指对象所属类在继承树中的深度(层次),其中树根为0。

DOI 的应用意义:①类的 DOI 越大,表示它的可能继承方法数目越大,预测它的行为将更困难。②继承树越深,设计越复杂,因为此树包括的方法和类越多。③一个特定类 DOI 越大,方法的重用越多。

Metric 3:孩子数目 NOC(Number Of Children)

定义3 NOC=继承树中的一个类的直接子孙的数目。

DOC 的应用意义:①NOC 越大,重用越好,因为继承也是一种形式重用。②NOC 越大,父类的不适当的抽象的可能性越大,因为一个类有很多子类,可能出现不正确的子类。③NOC 大表示该类在设计中有很大影响,此类应成为测试重点。

Metric 4:对象类之间耦合 CBO(Coupling Between Object classes)

定义4 一个类的 CBO 是它与别的类有耦合关系的类的数目。

CBO 应用意义:①对象类间过多的耦合对模块化设计和重用是有害的。一个类越独立,它就越易被另一个应用重用。②CBO 越大,设计中对另一部分的敏感越大,因此维护越困难。③CBO 对于不同设计部分的测试的复杂度估计是非常有用的。

Metric 5:一个类的响应集合 RFC (Response For a Class)

定义5 RFC=|RS|,RS 是该类的响应集合。

RFC 的应用意义:①RFC 越大,该类测试和调试将更复杂。②RFC 越大,类的复杂度越大。

Metric 6:类内聚缺乏度 LCOM(Lack of Cohesion in Methods)

定义6 类 C_i 有 n 个方法 $M_1 \cdots M_n$。设 $\{I_i\}$=被方法 M_i 使用的实例变量集合,$P=\{(I_i,I_j)\mid I_i \cap I_j=\varnothing\}$,$Q=\{(I_i,I_j)\mid I_i \cap I_j \neq \varnothing\}$,则

$$LCOM=\begin{cases}|P|-|Q|, & \text{当 } |P|>|Q| \text{ 时}\\ 0, & \text{其它}\end{cases}$$

如果 $I_i \cap I_j=\varnothing$,方法 M_i 和 M_j 为无关方法对,否则为相关方法对。

LCOM 的应用意义:①设计中希望类的内聚度越大越好,因为这样可以提高封装度。②LCOM 越大,意味着类可以分裂成两个或更多的子类。③LCOM 越大,会增加类复杂度、在开发过程中出错的可能性加大。

四、进一步讨论

传统的软件度量学发展只有短短20余年。虽然它取得了不少成果,但是度量学理论还缺乏坚实的理论基础。OO 开发技术具有更好的可靠性和可重用性,但由于它是新的方法,尚没有成熟。面向对象设计中的软件度量学的发展则只有几年时间,理论和方法有待于进一步发展,Chidamber 和 Kemerer 理论和方法还存在以下一些问题:

1)OO 术语的定义不够精确,Churcher 和 Shepperd 在[6]中尖锐地指出这一问题。Hitz 和 Montazeri 在[7]中指出 CBO 的度量没有考虑到各种耦合关系种类及它们强弱对 CBO 的影响。

2)某些度量在实际应用中难以达到设计时的目的。例如,LCOM 越小,内聚性越好。但从定义中不能认为 LCOM=0时,被测类的内聚性好。弓惠生在[5]中提出 LCOM 的计算方法:LCOM=(2|P|)/(n(n-1))这样0≤LCOM≥1,度量较为合理。

3)作者认为 Chidamber 和 Kermerer 提出的度量基于继承树这一模型,只涉及到类的度量,面向对象程序级的研究尚未涉及,需要进一步研究。

的确,这门新兴的学科发展的路还很长,需要解决的问题也还很多。但正如 Hitz 和 Montazeri[7]指出的"我们正在走向一条正确的路"。

参考文献

[1] R. Chidamber, F. Kemerer, Towards a Metrics Suite for Object-Oriented Design, OOPSLA'91

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

[3] R. Chidamber, F. Kemerer, Object-Oriented and Conventional Analysis and Design Methodologies: Comparison and Critique, IEEE Comput. Vol. 25, 1992

[4] N. E. Fenton, Software Metrics, A Rigorous Approach. New York: Chapman & Hall, 1991

[5] 弓惠生,面向对象设计中软件度量学的进展,计算机科学,23(3)1996

[6] Neville I. Churcher and Martin J. Shepperd, Towards a Conceptual Framework for Object-Oriented Software Metrics, ACM Softw. Eng. Notes, 20(2)1995

[7] Martin Hitz and Behzad Montazeri, R. Chidamber, F. Kemerer's Metrics Suite: A Measurement Theory Perspective, IEEE Trans., Softw., Eng., 22(4) 1996

[8] F. Roberts, Encyclopedia of Mathematics and its Applications. Reading, MA: Addision-Wesley, 1979