一种基于 UML 的 Java 类复杂性度量方法*

肖 汉[1,2]

(1. 郑州师范高等专科学校, 河南 郑州 450044; 2. 北京邮电大学 软件学院, 北京 100876)

摘 要: 软件产品复杂度在软件度量领域是一个非常重要的方面。软件开发的成本、进度和潜伏的错误数量均与复杂度有密切的关联。由于 OOA 和 OOD 被广泛运用,度量面向对象的软件复杂性成为必然的需求。UML 是一种面向对象分析与设计的建模语言,已经被接受为工业标准。为此,提出了一种适用于 Java 语言的根据 UML 类图对类的复杂性进行度量的方法,并用 Weyuker 提出的复杂性度量的评价标准对其进行了评价。

关键词: 复杂性度量; 面向对象; 统一建模语言; Java

中图分类号: TP311 **文献标志码**: A **文章编号**: 1001-3695(2007)07-0180-03

Metrics Suite for Java Class Complexity Based on UML

XIAO Han[1,2]

(1. *Zhengzhou Teacher's College, Zhengzhou Henan* 450044, *China*; 2. *School of Software, Beijing University of Posts & Telecommunications, Beijing* 100876, *China*)

Abstract: The complexity of software products is an important aspect in software measurement field. The complexity has a close relationship with the developing cost, time spending and the number of detects which a program may exist. OOA and OOD had been widely used, so the requirement of measuring software complexity written in object-oriented language is emerging. UML is a modeling language for OOA and OOD, it had been accepted as an industry standard. Then, this paper presented a suite of class complexity metrics based on UML class diagram that was adapted to Java, and verifies them with a suite of evaluation rules suggested by Weyuker.

Key words: complexity measurement; object-oriented; UML; Java

0 引言

20 世纪 80 年代后,面向对象方法和技术已受到计算机领域的专家、学者、研究人员和工程人员越来越广泛的重视。相继出现的一系列描述能力强、执行效率高的编程语言,标志着面向对象的方法与技术开始走向实用。90 年代以来,面向对象的开发方法学研究越来越活跃,各种成果大量涌现并被应用于实际开发中,特别是 Visual C++、Delphi、Java 等功能更强、效率更高、使用方便的面向对象语言的出现,使得面向对象方法和技术成为主流。

软件开发过程是高度智力活动,如何高质量、高效地开发软件,不仅是软件开发商关心的,也是从事软件开发技术与方法研究的专家、学者努力的目标。软件度量理论、方法的发展正是顺应了这种要求。

统一建模语言(UML)由 Rational 公司提出[1],它结合了 Rumbaugh、Booch 和 Jacobson 提出的三种面向对象分析与设计方法的优点,被 OMG 接受而成为事实上的面向对象建模语言的工业标准,并得到了广泛的应用。在软件的分析和设计阶段利用 UML 对类的复杂性进行度量,这种早期的度量可以给项目开发带来一些有用的信息,避免在软件开发的后期对软件的开发方案和设计原则进行重大调整。由于 Internet 的崛起和普及,Internet 上的网际程序语言 Java 已颇受瞩目。Java 是一种具备有异质平台执行能力、多线程,且支持网络、WWW 与图形用户界面的纯粹面向对象编程语言。由于它的许多先进功能而被视为是 WWW 上的最佳网际用语言之一。以 Java 做 WWW 上系统开发工具已经成为发展的趋势[2]。由于 Java 具有典型的面向对象的特征,选择 Java 语言作为研究对象,开发针对 Java 语言的软件度量方法既有代表意义又有实际价值。

1 面向对象度量的一些相关工作

面向对象的软件开发方法与传统的软件开发方法之间存在较大的差异,因此要根据面向对象系统的特点来定义新的度量准则,或对传统的软件度量准则进行扩展以适应面向对象系统的度量。许多研究人员对面向对象的度量方法进行了研究。Morris[3] 最早提出了对面向对象软件开发度量的一些度量准则。但这些准则没有经过验证。D. R. Morea 等人[4] 提出了对面向对象图形信息系统进行度量的三个度量准则,缺点是他们没有提供形式化和可测试的定义。S. R. Chidamber 等人[5] 提出了对面向对象设计进行度量的六个准则,它们是:①每个类的方法权重和(Weighted Methods per Class, WMC)。类中所有方法的复杂性之和。②继承树的深度(Depth of Inheritance Tree, DIT),即类的继承树的深度。在多重继承的情况下,它是从叶子节点到根节点的路径最大长度。③子类的个数(Num-

收稿日期: 2006-07-06; 修返日期: 2007-05-23 基金项目: 国家"863"计划资助项目(2001AA132011)

作者简介: 肖汉(1970-),男,湖北武汉人,副教授,CCF 会员,主要研究方向为软件工程、软件开发环境与开发技术(xiaohan70@163.com).

ber of Children, NOC) 。在继承的情况下,一个类的直接子类的个数。④类耦合度(Coupling Between Object classes, CBO) 。某一个类与其他类之间的耦合计数。⑤类的响应度(Response For a Class, RFC) 。接受外部的消息而在类中激发的方法数。⑥类内方法间的聚合缺乏度(Lack of Cohesion in Methods, LCOM) 。采用了两个项目中的类库,利用该度量方法得到一些经验数据。V. L. Basili 等人[6] 对 Chidamber 的度量方法进行了验证。Li[7] 指出了 Chidamber 度量准则中的一些缺点,并提出了另外一套度量准则:①祖先类的个数(Number of Ancestor Classes, NAC) 。一个类所继承的全部祖先类的数目。②子孙类的个数(Number of Descendent Classes, NDC) 。一个类在继承树中所有子孙类的个数。③类中局部方法的个数(Number of Local Methods, NLM) 。在类中定义的且可由外部访问的方法数(即可见性为 Public 的方法数) 。④类方法的复杂性(Class Method Complexity, CMC) 。类中所有方法的结构复杂性之和。⑤抽象数据类型之间的耦合(Coupling through Abstract data Type, CAT) 。在类的数据成员申明中作为抽象数据类型被使用的类的个数。⑥消息传递之间的耦合(Coupling Through Message passing, CTM) 。从类中发送出的不同类型的消息减去传送到在类的内部生成的对象消息。

2　基于 UML 的类复杂性度量

UML 是一个对面向对象分析和设计方法进行建模的描述语言。它包括用例的描述、类的静态描述、类间关系的静态描述、类的动态描述、类之间交互的动态描述以及系统描述等各个组成部分。这里主要讨论对类的复杂性进行度量,主要关注 UML 中的类图。

UML 中的类图主要是对类进行静态的描述,它可以描述 Java 语言中类的属性以及方法。为了正确地实现 Java 语言的度量,必须对 Java 语言文法进行深入研究分析。在对 Java 语言属性、方法和类的文法充分分析的基础上,从中可以导出两个度量准则(图 1) :①类中属性的个数; ②类中方法的个数。

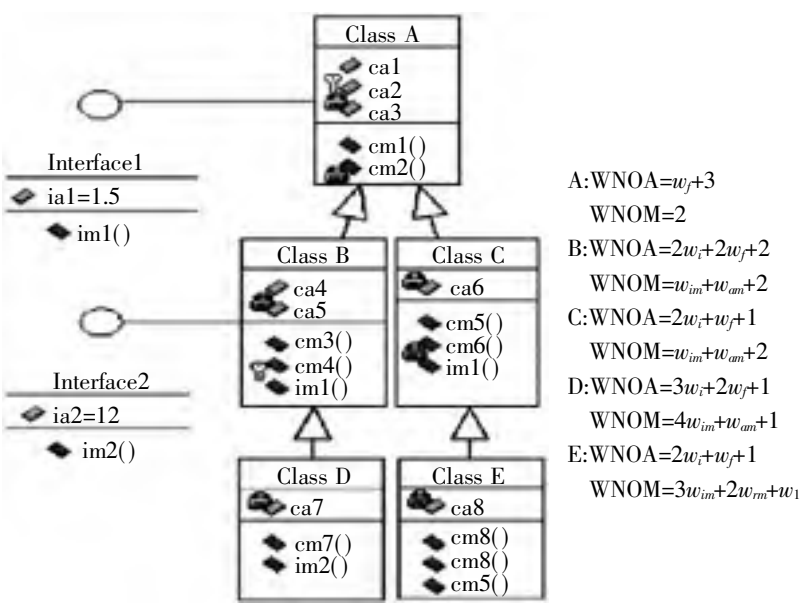

图 1　UML 类图的 Java 例子以及其属性和方法的加权值

度量准则 1　类中属性的加权值(Weighted Number of Attribute, WNOA)

定义　类的属性的加权值被定义为从祖先类中继承的属性的加权值、实现祖先接口时接口中定义的常量属性的加权值和各类特有的属性个数之和。

$$\mathrm{WNOA} = w_i p + w_f r + q$$

式中, p 为继承的属性数目; r 为接口实现时的属性数目; q 是该类中特有的属性数目。w_i 为继承的属性对类复杂性的贡献的权重; w_f 为接口实现时的属性对类复杂性的贡献的权重。w_i、w_f 应该取小于 1 的正数。

注意: 在类的继承关系中,祖先类的可见性为 Public 和 Protect的属性在子类中可用,而可见性为 Private 的属性在子类中不可用,因此 p 只包括那些可见性为 Public 和 Protect 的属性。祖先类包括该类的直接父类和祖先父类。祖先接口包括该类的直接父接口和祖先父接口。

度量准则 2　类中方法的加权值(Weighted Number of Method, WNOM)

定义　WNOM 为从父类中继承的方法的加权值和该类中特有的方法个数之和。

$$\mathrm{WNOM} = w_{im} Im + w_{am} Am + w_{rm} Rm + w_{lm} Lm + m$$

式中, Im 是从祖先类中继承的方法的个数; Am 是该类要实现的祖先接口中的抽象方法的个数,抽象方法必须要在子类中实现; Rm 是该类中重载同名方法的个数; Lm 是该类中覆盖祖先类中同名方法的个数; m 是该类中新添的方法的个数。w_{im} 是继承的方法对类复杂性的贡献的权重; w_{am} 是接口中的抽象方法对类复杂性的贡献的权重; w_{rm} 是重载方法对类复杂性的贡献的权重; w_{lm} 是覆盖方法对类复杂性的贡献的权重。w_{im}、w_{am}、w_{rm} 和 w_{lm} 是小于 1 的正数,并且 $w_{im} < w_{rm} < w_{lm} < w_{am}$。

注意: 在类的继承关系中,祖先类的可见性为 Public 和 Protect的方法在子类中可用,而可见性为 Private 的方法在子类中不可用,因此 Im 只包括那些可见性为 Public 和 Protect 的方法。祖先类包括该类的直接父类和祖先父类。祖先接口包括该类的直接父接口和祖先父接口。

这两个数值越大表明类越复杂。根据人们的直觉来看,某一事物的特性越多,所拥有的功能越多,它也就越复杂,这两个度量准则与人们的直觉观念相一致。

3　对以上度量准则的评价

E. J. Weyuker[8] 提出了对复杂性度量进行评价的九个特性。在这些特性被提出来之后,有许多研究人员对其提出了批评, N. E. Fenton[9] 提出 Weyuker 的特性不能确定对复杂性有一个单一的确定的认识。Zuse 批评 Weyuker 提出的特性与尺度理论不一致[10] 。Cherniavsky 等人提出 Weyuker 的特性只是一个好的度量准则所具有的必要条件而不是充分条件[5] 。但 E. J. Weyuker 的形式化的分析方法包容了以前非正规的对复杂性度量进行评价的特性,并为对复杂性度量进行评价提供了一定的理论基础。在文中仍然采用 Weyuker 的评价特性对提出的度量准则进行评价。S. R. Chidamber 等人[5] 提出在评价面向对象的复杂性度量时,要对 Weyuker 提出的度量准则进行调查,从而给出以下六个评价特性:

(1) 给定类 P 和 Q 以及度量准则 f, 存在 $f(P) \neq f(Q)$ 。它指出对于一个度量准则来说,并不是每一个类的复杂性度量均

产生相同的结果,否则将失去度量的意义。

(2) 存在不同的类 P 和 Q,使 $f(P) = f(Q)$。

(3) 给定两个类设计,即 P 和 Q。它们均提供相同的功能,但并不能因此得出 $f(P) = f(Q)$,即设计的细节信息会对度量结果产生影响。

(4) 对任意的类 P 和 Q,必须有: $f(P) \leq f(P+Q)$ 和 $f(Q) \leq f(P+Q)$ ($P+Q$ 指 P 和 Q 的合并)。它指出两个类的合并之后的类复杂性不应小于其中任一个类的复杂性。

(5) $\exists P$、$\exists Q$、$\exists R$,若 $f(P) = f(Q)$ 并不一定能得到 $f(P+R) = f(Q+R)$。这一特性指出 P 与 R 之间的交互和 Q 与 R 之间的交互可能会产生不同的结果,导致其复杂性也不一样。

(6) $\exists P$、$\exists Q$,使 $f(P) + f(Q) < f(P+Q)$。这一特性指出若 P 与 Q 合并后,它们之间的交互可能会使复杂度提高。

对于特性(1)和(2)来说,文中提出的两个度量准则很明显均能满足。在下面的讨论中不对这两个特性进行讨论,即对文中提出的复杂性度量准则进行评价。

度量准则 1 类中属性的个数 (Weighted Number of Attribute, WNOA)

对于特性(3),假定 P 和 Q 是两个完成相同功能的类,但其属性可能会不一致,比如说某一属性并不是类的属性,而是为了完成某一功能所加入的变量(如信号量等);而另一个完成相同功能的类则不使用这样的方式来完成相应的功能。它们两者的度量结果可能会产生差异,因此度量准则 1 满足这一特性。

对于特性(4),将 P 与 Q 进行合并得到一新类,则其属性的个数不会小于合并前的两个类。度量准则 1 满足这一特性。

对于特性(5),假定现有三个类 P、Q、R, $WNOA(P) = WNOA(Q)$,若 P 与 R 的属性的交集不为空,而 Q 与 R 的属性的交集为空,则会产生 $WNOA(P+R) < WNOA(Q+R)$ 的情况,所以该特性被满足。

对于特性(6),该特性得不到满足。对于 P 和 Q 的并,令 $I(X)$ 代表类的属性,若 $I(P) \cap I(Q) = \emptyset$,则 $WNOA(P) + WNOA(Q) = WNOA(P+Q)$,若 $I(P) \cap I(Q) \neq \emptyset$,则 $WNOA(P) + WNOA(Q) > WNOA(P+Q)$。

度量准则 2 类中方法的个数(Weighted Number of Method, WNOM)

对于特性(3),假定 P 和 Q 是两个完成相同功能的类,但其方法可能会不一致,如若 P 类中的方法 A 调用了方法 B 从而完成某一个操作;但在类 Q 中为了完成相同类型的操作,可能不把 B 设计为方法,而是将它放在了方法 A 中,这样虽然它们实现了相同的功能,但两者的度量结果可能会产生差异。因此度量准则 2 满足这一特性。

对于特性(4),将 P 与 Q 进行合并得到一新类,则其方法的个数不会小于合并前的两个类,度量准则 2 满足这一特性。

对于特性(5),假定现有三个类 P、Q、R, $WNOM(P) = WNOM(Q)$,若 P 与 R 的方法的交集不为空,而 Q 与 R 的方法的交集为空,则会产生 $WNOM(P+R) < WNOM(Q+R)$ 的情况,所以该特性被满足。

对于特性(6),该特性得不到满足。因为对于 P 与 Q 的并,令 $M(X)$ 代表类的方法,若 $M(P) \cap M(Q) = \quad$,则 $WNOM(P) + WNOM(Q) = WNOM(P+Q)$;若 $M(P)\ M(Q) \neq \quad$,则 $WNOM(P) + WNOM(Q) > WNOM(P+Q)$。

从以上的讨论可以看出,度量准则 1 和 2 不满足特性(6)。对于特性(6)来说,其出发点是考虑类之间的交互可能会导致其复杂性的提高,它着重强调类之间的动态特性,但度量准则 1 和 2 度量的是类的静态特性,因此这种不满足的情况是合理的。

4 结束语

在对 Java 语言文法作了详细的分析以后,本文提出了两个根据 UML 对 Java 类的复杂性进行度量的度量准则,并用 Weyuker 提出的复杂性度量的评价标准对其进行了评价。该方法应用于面向对象分析和设计阶段,并可与某些利用 UML 的 Case 工具相集成,在设计阶段自动给出度量的结果供设计人员或项目管理人员在进行决策时使用。UML 得到了工业界的认可并被广泛使用,因此这套度量方法的应用前景十分广阔。本文仅讨论了对类的复杂性度量的准则。对于某些类组合成的子系统或系统的复杂性并没有进行讨论。由于类之间存在各种各样的动态关系(消息传递、激活等),在以后的工作中可以进一步对类间的动态关系进行研究,归纳出一定的度量准则,使该理论能得到进一步的补充和完备。另外度量准则 1 和 2 中的各个权值文中并没有给出,这些权值的确定需要大量的统计和调查工作,这也是今后所要解决的问题之一。

参考文献:

[1] ML Proposal to the Object Management Group. version 1.1, Rational corporation[EB/OL]. [1997-09]. http://www.rational.om/uml.

[2] 邢大红,李家伟,张顺国,等. Java 语言的度量及工具实现[J]. 计算机工程, 2001, 27(9): 51-52.

[3] MORRIS K. Metrics for object-oriented software development[D]. Cambridge, MA: MIT Sloan school of Management, 1988.

[4] MOEAU D R, DOMINICK W D. Object oriented graphical information systems: research plan and evaluation metrics[J]. The Journal of Systems and Software, 1989, 10: 23-28.

[5] CHIDAMBER S R, KEMERER C F. A metrics suite for object oriented design[J]. IEEE Transaction on Software Engineering, 1994, 20(6): 476-493.

[6] BASILI V L, BRIAND L, MELO W L. A validation of object-oriented metrics as quality indicators[J]. IEEE Transaction on Software Engineering, 1988, 10: 751-761.

[7] LI Wei. Another metrics suite for object-oriented programming[J]. The Journal of System and Software, 1998, 44: 155-162.

[8] WEYUKER E J. Evaluating software complexity measures[J]. IEEE Transaction on Software Engineering, 1988, 14(9): 1357-1365.

[9] FENTON N E. Software metrics: a rigorous approach[M]. New York: Chapman & Hall, 1991.

[10] 伦立军,丁雪梅,李英梅. 程序复杂性度量的一种新方法[J]. 计算机应用研究, 2004, 21(2): 37-39.