《面向对象软件度量阈值确定方法研究问题、进展与挑战》附录

**Table A1** Definition and literature sources of object-oriented software metrics——cohesion

表A**1** 面向对象软件度量定义和文献来源——内聚性(32个)

|  |  |  |  |
| --- | --- | --- | --- |
| 度量名 | 年份 | 定义 | 文献来源 |
| ***LCOM1*** | 1991 | ***Lack of cohesion in methods.*** 某个类的方法中缺少内聚性. 即某个类中没有使用共同属性的方法数量. | Chidamber等[1] |
| ***LCOM2*** | 1994 | 某个类没有使用共同属性的方法数量减去该类使用共同属性的方法数量,若为负数, 则取值为零. | Chidamber等[2] |
| ***LCOM3*** | 1995 | 考虑一个无向图G, 其中顶点是类的方法数量, 如果两个方法使用至少一个共同的属性, 则两个顶点之间有一条边. LCOM3定义为G的连接组件的数量. | Hitz等[3] |
| ***LCOM4*** | 1995 | 与*LCOM3*类似. 当某个类中, 方法1调用方法2, 则无向图G增加了两个顶点的一条边来表示方法1与方法2, 反之亦然. | Briand等[4] |
| ***LCOM5*** | 1996 | 考虑一个方法数据集，并得到一个属性数据集。令等于引用属性的方法数量，则. | Henderson等[5] |
| ***Co*** | 1998 | ***Connectivity.*** 连通性. 设是LCOM4中图的顶点数据集, 是图边的数据集, 则. | Briand等[4] |
| ***NewCo*** | 1998 | Co的变体. | Briand等[4] |
| ***TCC*** | 1995 | ***Tight class cohesion.*** 类的内聚性紧密程度, 包括直接使用属性的方法和间接使用属性的方法. 如果方法m直接或间接地调用一个直接使用属性a的方法, 则称方法m间接调用属性a. 如果两个方法直接或间接使用共同的属性，则称这两个方法是连通的, 其数量用表示. 设是直接连通数量. TCC则是公共的方法中连通对数的比例, . 例如, 直接或间接使用公共属性方法的对数. | Bieman等人[6] |
| ***LCC*** | 1995 | ***Loose class cohesion.*** 类的内聚性松散程度. 与TCC类似, 但要除去该度量也考虑的间接连通的方法数量. 如果某个类有方法, 例如与是连通的 ，那么与是间接连通的. 设是间接连通数量. LCC则是某个类的公共方法中直接或间接连通的对数的比例, . | Bieman等[6] |
| ***ICH*** | 1995 | ***Information flow through method invocations within a class.***信息流为基础的内聚. 某个类的方法ICH值定义为对同一个类中其他方法的调用次数, 并以被调用方法的参数数目为权重. 一个类的ICH是其方法ICH值的总和。 | Lee等[7] |
| ***OCC*** | 2002 | ***Optimistic Class Cohesion.*** 考虑一个无向图G, 其中顶点数量是某个类的方法数量, 如果两个方法直接或间接至少引用一个公共属性，则两个顶点之间的存在一条边. 设N是某个类的方法总数量, 那么, OCC是度量从方法 到其它方法的可到达数量的最高比率, 该比率最低取0. | Aman等[8] |
| ***PCC*** | 2002 | ***Pessimistic Class Cohesion.*** 与OCC类似，但除去直接的无向图。如果一个顶点的方法直接或间接写入某个属性与另一个顶点的方法直接或间接读入该属性, 则两个顶点之间有一条弧. | Aman等[8] |
| ***DCD*** | 2004 | ***Degree of cohesion* (*based on the direct relation between its public methods*)*.*** 直接相关公共方法对(pairs)的百分比. 与TCC类似. 但要除去两个方法连通的情况: 如一个方法调用另一个方法; 两个方法直接或间接使用共同的属性和调用公共的方法. 考虑一个无向图, 其中顶点是类的公共方法的数量. 公共方法对的最大数量为. 如果两个方法直接相关, 则两个顶点之间存在一条边. 设是图中的边数, 则. | Badri等[9] |
| ***LCCD*** | 2004 | ***Lack of Cohesion in the Class*(*the direct relation*)*.*** 类中缺少的内聚性(直接相关的公共方法). . | Badri等[9] |
| ***DCI*** | 2004 | ***Degree of cohesion* (*based on the direct and indirect relations between its public methods*)*.*** 直接或间接相关公共方法对(pairs)的百分比. 与LCC类似. 但要除去两个方法连通的情况: 如一个方法调用另一个方法; 两个方法直接或间接使用共同的属性和调用公共的方法. 考虑一个无向图, 其中顶点是类的公共方法的数量. 如果两个方法直接或间接相关, 则两个顶点之间存在一条边. 设是图中的边数, 则. | Badri等[9] |
| ***LCCI*** | 2004 | ***Lack of Cohesion in the Class*(*the direct and indirect relations*)*.*** 类中缺少的内聚性(直接或间接相关的公共方法). . | Badri等[9] |
| ***CAMC*** | 1999 | ***Cohesion Among Methods of Classes.*** 类中的方法之间的内聚性. 考虑一个 的参数发生矩阵*PO*, 是某个类的方法数量, 是这些方法不同参数类型的数量. 元素的位置, 当在第个方法中发生了第个数据类型, 则取1, 反之取0, , . 则 . | Bansiya等[10] |
| ***NHD*** | 2006 | ***Normalised Hamming distance based cohesion.*** 标准化的Hamming距离为基础的内聚性. 设是方法与方法的参数协议, 它等于两个方法中参数发生矩阵对应位置上数字相等的数量. NHD是某个类中所有方法之间参数协议平均值的标准化. 即 . | Counsell等[11] |
| ***SNHD*** | 2006 | ***Scaled NHD.*** 规模化的NHD数值. | Counsell等[11] |
| ***RCI*** | 1994 | ***Ratio of Cohesive Interactions.*** 内聚性相互作用比率. 类可被为数据声明和方法的集合. 数据声明包括: 局部和公共类型声明; 类; 公共属性(包括常量). 若数据声明a或a在应用中的一个改变导致另一个数据声明b或b在应用中的改变, 则称a与b之间存在数据声明相互作用(DD-*interaction*). 若数据声明a与至少一个方法m的数据声明发生数据声明相互作用(DD-*interaction*), 则称数据声明a与该方法m之间存在数据声明与方法之间相互作用(DM-*interaction*). 设(cohesive interactions)是所有DD-和DM-*interaction*的集合. 是类接口中所有可能的DD和DM-*interaction*的集合. 对所有的类, .. 0和1分别表示最小和最大的内聚性. | Briand等[12] [13] [4] |
| ***NRCI*** | 1998 | ***The neutral ratio of cohesive interactions.*** 内聚性相互作用的中性比.若表示所有已知的相互作用, 则. 在开发阶段, 由于可用信息不够, 某些交互作用可能存在或可能不存在. 表示这些所有未知的相互作用. 同时, 其余的交互作用被认为不存在. 在不考虑未知的相互作用时,. | Briand等[4] |
| ***PRCI*** | 1998 | ***The pessimistic ratio of cohesive interactions.*** 内聚相互作用的悲观比例. 当未知的交互作用被认为不是真实的交互作用时,. | Briand等[4] |
| ***ORCI*** | 1998 | ***The optimistic ratio of cohesive interactions.***  内聚相互作用的乐观比例. 当未知的交互作用被认为是真实的交互作用时,. | Briand等[4] |
| ***DAM*** | 2002 | ***Data Access Metric.*** 数据访问度量. 该度量值等于类中私有(受保护的)的属性数量与类中所有声明的属性总数的比率. 其取值范围[0, 1]. 其数值越大越好. | Bansiya等[14] |
| ***CAM*** | 2002 | ***Cohesion Among Methods of Class.*** 类方法内聚性. 该度量基于方法的参数列表计算某个类方法间的相关性. 其数值等于使用每种方法中不同类型的方法参数数量的总和除以整个类中不同方法参数类型的数量和方法数量的乘积. 其取值范围[0, 1], 数值越大越好. | Bansiya等[14] |
| ***MOA*** | 2002 | ***Measure of Aggregation.*** 聚合性度量. 该度量通过使用属性来实现部分与整体关系的程度, 其数值等于对用户定义类中字段(class fields)计数. | Bansiya等[14] |
| ***CBMC*** | 1998 | ***Cohesion Based on Member Connectivity.*** 基于类中成员连通性内聚.不仅要考虑交互的数量，还要考虑类中方法之间的交互模式。该度量考虑了粘合方法(glue methods)的数量与交互方法数量的比率。粘接方法的数量等于满足删除它们会导致方法-属性交互图变得不相交的条件下的最小数量. | Chae等[15] [16] |
| ***ICBMC*** | 2002 | ***An improved version of CBMC.*** 基于类中成员连通性内聚度量的提高版本. 它考虑的是切割集(cut sets)而不是粘合方法。切割集是去除这些边会使方法-属性交互图变得不相交条件下边的最小集. | Zhou等[17] |
| ***SCC*** | 2010 | ***Similarity-based Class Cohesion.*** 相似性为基础的内聚性. ***SCC***度量是考虑方法-方法、属性-属性和属性-方法直接交互和传递交互的四个不同度量的加权平均值。 | Al Dallal等[18] |
| ***SCOM*** | 2006 | ***Sensitive Class Cohesion Metric.*** 敏感的类内聚性度量. 所有方法对之间的相似性总和与方法对总数的比值. 设某个类有属性:和方法, 设表示在给定方法下属性(attributes)的子集, 设a是类中实例属性的总数. 因为两个集合交集最大可能的基数是它们的基数(cardinalities)的最小值, 则一对方法的连通强度(connection intensity)为:, 在考虑权重的情况下, . | Fernández等[19] |
| ***CC*** | 2006 | ***Class Cohesion (CC).*** 类内聚性度量. 所有方法对之间的相似性总和与方法对总数的比值. 其方法与方法的相似性(连通强度)定义为:  , 其中, 和分别是方法与方法的属性集合. | Bonja等[20] |
| ***LSCC*** | 2012 | ***LLD Similarity-based Class Cohesion. LLD: Low-level design.*** 低层次设计的相似性为基础的内聚性. *LSCC*度量是以两方法引用共同属性数/类属性总数定义相似性, 此相似性总和/方法对总数. | Al Dallal等[21] |

**Table A2** Definition and literature sources of object-oriented software metrics——coupling

表A**2** 面向对象软件度量定义和文献来源——耦合性(27个)

|  |  |  |  |
| --- | --- | --- | --- |
| 度量名 | 年份 | 定义 | 文献来源 |
| ***CBO*** | 1994 | ***Coupling between classes.*** 类之间的耦合性. 某个类与它的非继承相关类耦合的数量. 一个类与另一个类耦合是指某个类使用另一类的方法或属性. | Chidamber等[2] |
| ***RFC*** | 1994 | ***Response set for classes.*** 类的响应数据集. 某个类的响应数据集包括该类的方法数M和这些M个方法直接或间接调用方法数. RFC指类的响应数据集中的方法数量. | Chidamber等[2] |
| ***Ca*** | 1994 | ***Afferent Couplings.*** 传入耦合. 该度量表示依赖于所度量类的类数量. | Martin等[22] |
| ***Ce*** | 1994 | ***Efferent Couplings.*** 传出耦合. 该度量表示被度量类所依赖类的数量. | Martin等[22] |
| ***I*** | 1994 | ***Instability.*** 传入和传出耦合下的不稳定性. . 当I=0时,表示最大的稳定性; 当I=1时,表示最大的不稳定性. | Martin等[22] |
| ***MPC*** | 1993 | ***Message passing coupling.*** 消息传递的耦合性. 某一个类中方法调用的数量. | Li等[23] |
| ***DAC*** | 1993 | ***Data abstraction coupling.*** 数据抽象耦合性. 某一个类拥有与另一个类的相同属性类型的数量. | Li等[23] |
| ***ICP*** | 1995 | ***Information-flow-based coupling.*** 信息流的耦合. 某一个类中方法调用的数量, 且以方法调用参数的数量加权后得到. | Lee等[7] |
| ***IH-ICP*** | 1995 | 与ICP类似. 但只计算某个类祖先的若干类的方法调用数量(即基于继承的耦合). | Lee等[7] |
| ***NIH-ICP*** | 1995 | 与ICP类似, 但计算调用与继承类不相关类的方法数. | Lee等[7] |
| ***ACAIC***  ***ACMIC***  ***AMMIC***  ***DCAEC***  ***DCMEC***  ***DMMEC***  ***OCAEC***  ***OCAIC***  ***OCMEC***  ***OCMIC***  ***OMMEC***  ***OMMIC*** | 1997 | 这12个耦合度量是通过类之间的相互作用计算的. 对这些度量的缩写反应相互作用的计算过程:  1. 第一个字母表明相互关系: A表示与祖先类的耦合; D表示与派生类的耦合; O表示与其他类(如没有继承关系)的耦合.  2. 第二个至第三个字母表示相互作用的类型: CA表示类c与类d之间类与属性的交互过程, 即类c中有一个与类d相同类型的属性; CM表示类c与类d之间类与方法的交互过程, 即类c中有一个与类d相同参数类型的方法; MM表示表示类c与类d之间方法与方法的交互过程, 即类c调用类d的一个方法, 或类d的一个方法对类c的一个方法传递参数(函数指针).  3. 最后两个字母表示耦合影响的场所: IC表示内部耦合; EC表示外部耦合. | Briand等[24] [25] |
| ***CBI*** | 1996 | Coupling based on inheritance. 基于继承耦合. 它是某一个类内部所有方法执行的复杂度之和与其派生类数量的乘积. 某个类中的方法复杂度是由霍尔斯特德软件科学工作(Halstead’s software science)测量得到的。 | Kim等[26] |
| ***NAS*** | 1998 | Number of Associations. 类与同级之间的关联数量. 该度量直接从设计文档中收集数据, 采用自上而下的目标—问题度量(GQM)方法[27]收集. 它与CBO相同, 除非以下情况: 如果一个类使用另一个类的方法超过一次, CBO度量对每一次使用作为单独发生的耦合来计数; 而对于NAS, 一个类对另一个类的方法多次调用, 只算一次单独发生的耦合来计数. 因此, 它与CBO有很强的相关性, 但两者并不是正交的. | Harrison等[28] |
| ***DCC*** | 2002 | Direct Class Coupling. 类的直接耦合. 该度量对一个类直接耦合的不同类计数, 包括属性声明中直接相关和方法中信息传递的类. | Bansiya等[14] |
| ***IC*** | 1999 | Inheritance Coupling. 继承耦合数. 该度量表示与某个类耦合的父类数量. 当某个类的继承方法在功能上依赖于该类中新方法或重新定义的方法, 该类将与其父类耦合. 具体来说, 如果有一个继承方法满足以下条件之一, 则称一个类与其父类耦合: 使用一个新方法或重新定义的方法里的属性; 调用重新定义的方法; 被重新定义的方法调用; 和使用在重新定义方法中定义的参数. | Tang等[29] |
| ***CBM*** | 1999 | Coupling Between Methods. 方法间的耦合. 与所有继承方法都耦合的新方法或重新定义的方法总数. 当满足IC度量中至少一条耦合条件时, CBM度量的耦合条件被满足. | Tang等[29] |

**Table A3** Definition and literature sources of object-oriented software metrics——Inheritance

表A**3** 面向对象软件度量定义和文献来源——继承性(20个)

|  |  |  |  |
| --- | --- | --- | --- |
| 度量名 | 年份 | 定义 | 文献来源 |
| ***DIT*** | 1991 | ***Depth of inheritance tree.*** 树继承的深度. 某个类在整个继承的体系中到根结点的最长路径. | Chidamber  等[1] [2] |
| ***AID*** | 1996 | ***Average inheritance depth of a class.*** 某个类的平均继承深度. 若某个类没有祖先(父类), 则AID为零. 除此以外, 其他类的AID是该类若干父类的AID的平均数, 以1的幅度增加. | Henderson-Sellers等[30] |
| ***CLD*** | 1992 | ***Class-to-leaf depth.*** 类到叶子的深度. 在整个继承层次中, 某个类以下的若干子类中, 到叶子的最大数量. | Tegarden等[31] |
| ***NOC*** | 1991 | ***Number of children.*** 直接继承子类的数量. 直接从某个类继承的若干子类的数量. | Chidamber  等[2] [1] |
| ***NOP\**** | 1994 | ***Number of parents.*** 直接继承父类的数量. 某个类的直接继承的父类的数量. | Lorenz等[32]和  Lake等 [33] |
| ***NOD*** | 1992 | ***Number of descendents.*** 后代的数量. 直接或间接继承的子类数量. | Tegarden等[31]  和Lake等 [33] |
| ***NOA*** | 1992 | ***Number of ancestors.*** 祖先的数量. 某个类直接或间接从其父类继承的数量. | Tegarden等[31] |
| ***NMO*** | 1994 | ***Number of methods overridden.*** 方法被重写的数量. 某个类从祖先继承下来被重写的方法数量. | Lorenz等[32] |
| ***NMI*** | 1994 | ***Number of methods inherited.*** 方法被继承的数量. 某个类从祖先继承下来且没有被重写的方法数量. | Lorenz等[32] |
| ***NMA*** | 1994 | ***Number of methods added.*** 增加的方法数量. 某个类中不是继承的、没有重写的方法数量. | Lorenz等[32] |
| ***SIX*** | 1994 | ***Specialization index.*** 专业化指数. SIX = NMO \* DIT /(NMO+NMA+NMI) | Lorenz等[32] |
| ***SPA*** | 1999 | ***Static polymorphism in ancestors.*** 某个类祖先的静态多态性. 某个类及其祖先类中静态多态函数出现的次数. 静态多态函数是指在不同类中定义的函数, 并具有相同的名称和不同的特征(signatures). | Benlarbi等[34] |
| ***SPD*** | 1999 | ***Static polymorphism in descendants.*** 后代的静态多态性. 某个类及其子类静态多态函数出现的数量. | Benlarbi等[34] |
| ***SP*** | 1999 | ***Static polymorphism in inheritance relations.*** 继承关系的静态多态性.  SP = SPA+SPD. | Benlarbi等[34] |
| ***DPA*** | 1999 | ***Dynamic polymorphism in ancestors.*** 某个类祖先的动态多态性. 某个类及其祖先类中动态多态函数出现的次数. 动态多态函数是指在不同类中定义的函数, 并具有相同的名称和相同的特征(signatures). | Benlarbi等[34] |
| ***DPD*** | 1999 | ***Dynamic polymorphism in descendants.*** 后代的动态多态性. 某个类及其子类动态多态函数出现的数量. | Benlarbi等[34] |
| ***DP*** | 1999 | ***Dynamic polymorphism in inheritance relations.*** 继承关系的动态多态性.  DP = DPA+DPD. | Benlarbi等[34] |
| ***NOH*** | 2002 | ***Number of Hierarchies.*** 类层次数量. 该度量对设计阶段的类层次计数. | Bansiya等[14] |
| ***ANA*** | 2002 | ***Average Number of Ancestors.*** 平均父类数量. 该度量表示从某个类继承后类数量的平均值. 其数值等于在继承结构下从根类到所有类的所有路径上类的数量. | Bansiya等[14] |
| ***MFA*** | 2002 | ***Measure of Functional Abstraction.*** 函数抽象度量. 该度量是类继承的方法数量与该类的成员方法可访问的方法总数比率. 其取值范围[0, 1]. | Bansiya等[14] |

\*:与类规模NOP重名, 但定义不同.

**Table A4** Definition and literature sources of object-oriented software metrics——Size and complexity

表A**4** 面向对象软件度量定义和文献来源——规模和复杂性(31个)

|  |  |  |  |
| --- | --- | --- | --- |
| 度量名 | 年份 | 定义 | 文献来源 |
| ***CC*** | 1976 | ***McCabe’s cyclomatic complexity.*** McCabe的圈复杂度. 设图G有n个顶点, e条边和p个连通分量, 则圈复杂度. 其原理是: 在强连接图G中, 圈复杂度等于其线性独立的回路的最大数. | McCabe等[35] |
| ***maxCC*** | 1976 | ***Maximum CC.*** CC的最大值. 即某个类方法中CC的最大值. | McCabe等[35] |
| ***avgCC*** | 1976 | ***Average CC***. CC的平均值. 即某个类方法中CC的算术平均值. | McCabe等[35] |
| ***AC*** | 1976 | ***Actual Complexity Metric (ac).*** 实际的复杂性度量. 实际复杂度(ac)是测试期间遍历独立路径的数量。  ***Actual Complexity (ac) is the number of independent paths traversed during testing.*** | McCabe等[35] |
| ***E*** | 1977 | ***Halstead's effort metric(E).*** Halstead工作量(复杂度)度量. 设不同的运算符(operator)数量; 不同的操作数(operand)数量; 运算符(operator)出现总次数; 操作数(operand)出现总次数, 则. | Halstead等[36]和Curtis等[37] |
| ***WMC*** | 1991 | ***Weighted Methods Per Class.*** 类中加权的方法和, 表示一个类中所有方法的复杂度之和. 该CK度量未指明是用圈复杂度来衡量复杂度. 若某个类C有方法, 设为每个方法的静态复杂度, 则. 若所有的方法的静态复杂度都一致(unity), 则, 即WMC为方法数之和. | Chidamber等[1] |
| ***NOM*** | 1993 | ***Number of local methods.*** 类中方法数. 一个类中接口增量度量(interface increment metric). 由于类中的局部方法构成了类的接口增量，因此*NOM*可以作为最好的交互度量. | Li等[23] |
| ***SIZE1*** | 1993 | *SIZE1*=一个类中分号数量. 通过对一个类中分号计数来统计传统*LOC*度量值. | Li等[23] |
| ***SIZE2*** | 1993 | *SIZE2*=一个类中属性数量+类中方法数量. | Li等[23] |
| ***NA*** | 2000 | ***Number of attributes.*** 某个类的属性数量, 包括继承和非继承. | Briand等[38] |
| ***NAIMP*** | 2001 | ***Number of attributes* (*excluding inherited ones*)*.*** 某个类的属性数量, 不包括继承的类. | Briand等[39] |
| ***NM*** | 2000 | ***Number of methods* (*both inherited and noninherited*)*.*** 某个类的方法数量, 包括继承和非继承. | Briand等[38] |
| ***NMIMP*** | 2001 | ***Number of methods implemented.*** 某个类实现的方法数量. | Briand等[39] |
| ***NumPara*** | 2000 | ***Sum of the number of parameters of the methods implemented.*** 某个类实现的方法中参数的总数. | Briand等[38] |
| ***Stmts*** | 2000 | ***Number of declaration and executable statements in the methods.*** 某个类的方法中的声明和可执行语句的数量. | Briand等[38] |
| ***NPM*** | 2002 | ***Number of Public Methods.*** 公共方法的数量. 该度量仅对类中声明为公共的方法计数. | Bansiya等[14] |
| ***CIS*** | 2002 | ***Class Interface Size.*** 类的接口规模. 与NPM相同. | Bansiya等[14] |
| ***DSC*** | 2002 | ***Design Size in Classes.***设计阶段类规模. 该度量对设计阶段的类计数. | Bansiya等[14] |
| ***NOP\**** | 2002 | ***Number of Polymorphic Methods.*** 多态性方法数量. 即可以展现为多态行为的方法数量. 如C++中被标记为virtual的方法数量. | Bansiya等[14] |
| ***AMC*** | 1999 | ***Average Method Complexity.*** 方法的平均复杂度. 即表示每个类方法平均规模大小. 该度量不包括纯虚拟方法(pure virtual methods)和继承方法. 其背后的假设是规模大的方法比规模小的方法倾向于引入更多的缺陷. | Tang等[29] |
| ***NOMA*** | 1999 | ***Number of Object/Memory Allocation.*** 对象或内存分配数量. 表示一个类实例一个对象(分配内存)时语句(statements)的总数. 不包括调用其他方法的间接分配内存. | Tang等[29] |
| ***SLOC*** | 1977 | ***Noncommentary source lines of code.*** 某个类中非注释的源代码行数. | Halstead等[36] |
| ***FTF*** | 2005 | ***Function Template Factor.***使用函数模板的函数数量与所有函数的数量之比. | Aggarwal等[40] |
| ***CTF*** | 2005 | ***Class Template Factor.*** 使用类模板的类数量与所有类的数量之比. | Aggarwal等[40] |
| ***AA*** | 2018 | ***Attributes in Annotations.*** 属性注释度量. 在一个特定的注释声明中有多少属性. | Lima等[41] |
| ***LOCAD*** | 2018 | ***LOC in Annotation Declaration.*** 注释声明的*LOC*度量. *LOCAD*度量用于完全声明注释的行数. 过多代码行的注释声明会损害其可读性、维护和发展。通常，具有较高*LOCAD*的注释也有很多属性(用AA来度量). 但是，它们不是一对一的关系, 因为一行注释可以有多个属性. | Lima等[41] |
| ***ANL*** | 2018 | ***Annotation Nesting Level.*** 注释嵌套层次度量. *ANL*度量注释嵌套层次的深度. 嵌套层次通常非常低，因为某些类型的注释使用其他注释作为属性, 所以大多数值都是零. | Lima等[41] |
| ***AED*** | 2018 | ***Annotations in Element Declaration.* 成员声明的注释.** 源代码成员, 如方法、属性和类, 一般都可以进行注释. *AED*度量各成员声明中的注释数量, 包括其嵌套的注释. 同一个成员中大量的注释可能会导致代码难以维护. 拥有过多注释的成员可能会在不破坏其他部分的情况下阻止代码的发展. | Lima等[41] |
| ***AC*** | 2018 | ***Annotations in Class.*** 类的注释. *AC*度量一个类包含注释的数量. 由于注释视为可选特性, 所以并不是所有的类都包含它. 但是, 一些类可能包含大量注释, 能达到500多个. | Lima等[41] |
| ***UAC*** | 2018 | ***Unique Annotations in Class.*** 类的独特注释. 与*AC*不同，*UAC*关注的是类中不同注释的数量. 如果一个注释具有相同的类型和相同的属性值，则认为它与另一个注释是等效的. 从*AC*中减去*UAC*就得到了重复注释的数量. 根据定义，类的*UAC*值永远不会大于*AC*值. | Lima等[41] |
| ***ASC*** | 2018 | ***Annotations Schemas in Class.*** 类的注释模式.  当软件在其成员中使用注释时, 它们属于表示给定值域下元数据的集合, 称为模式(*Schema*). 当应用程序使用基于元数据的框架时, 它通常使用来自该框架的一组注释来表示必要的元数据. 应用程序可以根据需要使用任意多的模式. *ASC*度量模式的数量, 可以由注释包标识. | Lima等[41] |

\*:与继承性NOP重名, 但定义不同.

**参考文献：**

[1] Chidamber SR, Kemerer CF. Towards a metrics suite for object oriented design. Conference proceedings on Object-oriented programming systems, languages, and applications, 1991: 197-211.[doi: 10.1145/118014.117970]

[2] Chidamber SR, Kemerer CF. A metrics suite for object oriented design. IEEE Transactions on software engineering, 1994,20(6):476-493.[doi: 10.1109/32.295895]

[3] Hitz M, Montazeri B. Measuring coupling and cohesion in object-oriented systems. Proc. Int. Symposium on Applied Corporate Computing, 1995.[doi: 10.1.1.409.4862]

[4] Briand LC, Daly JW, Wüst J. A unified framework for cohesion measurement in object-oriented systems. Empirical Software Engineering, 1998,3(1):65-117.[doi: 10.1023/A:1009783721306]

[5] Henderson-Sellers B. Software metrics. Prentice Hall, Hemel Hempstaed, UK, 1996

[6] Bieman JM, Kang B-K. Cohesion and reuse in an object-oriented system. ACM SIGSOFT Software Engineering Notes, 1995,20(SI):259-262.[doi: 10.1145/223427.211856]

[7] Lee Y-s, Liang B. Measuring the coupling and cohesion of an object-oriented program based on information flow. Proceedings of the International Conference on Software Quality, 1995: 81-90

[8] Aman H, Yamasaki K, Yamada H, Noda M-T. A proposal of class cohesion metrics using sizes of cohesive parts. Proc. of Fifth Joint Conference on Knowledge-based Software Engineering, 2002: 102-107

[9] Badri L, Badri M. A Proposal of a new class cohesion criterion: an empirical study. Journal of Object Technology, 2004,3(4):145-159.[doi: 10.5381/jot.2004.3.4.a8]

[10] Bansiya J. Class Cohesion Metric For Object Oriented Designs. Journal of Object-Oriented Programming, 1999,11(8):47-52

[11] Counsell S, Swift S, Crampton J. The interpretation and utility of three cohesion metrics for object-oriented design. ACM Transactions on Software Engineering and Methodology (TOSEM), 2006,15(2):123-149.[doi: 10.1145/1131421.1131422]

[12] Briand LC, Morasca S, Basili VR. Measuring and assessing maintainability at the end of high level design. 1993 Conference on Software Maintenance, 1993: 88-87.[doi: 10.1109/ICSM.1993.366952]

[13] Briand L, Morasca S, Basili VR. Defining and validating high-level design metrics, 1994.[doi: 10.1.1.31.4744]

[14] Bansiya J, Davis CG. A hierarchical model for object-oriented design quality assessment. IEEE Transactions on software engineering, 2002,28(1):4-17.[doi: 10.1109/32.979986]

[15] Chae H, Kwon Y. A cohesion measure for classes in object-oriented systems. Proceedings Fifth International Software Metrics Symposium, 1998: 158-166.[doi: 10.1109/METRIC.1998.731241]

[16] Chae HS, Kwon YR, Bae DH. A cohesion measure for object‐oriented classes. Software: Practice and Experience, 2000,30(12):1405-1431.[doi: 10.1002/1097-024X(200010)30:123.0.CO;2-3]

[17] Zhou Y, Xu B, Zhao J, Yang H. ICBMC: an improved cohesion measure for classes. International Conference on Software Maintenance, 2002. Proceedings., 2002: 44-53.[doi: 10.1109/ICSM.2002.1167746]

[18] Al Dallal J. Mathematical validation of object-oriented class cohesion metrics. International Journal of Computers, 2010,4(2):45-52

[19] Fernández L, Peña R. A sensitive metric of class cohesion. International Journal "Information Theories & Applications", 2006,13(1):82-91

[20] Bonja C, Kidanmariam E. Metrics for class cohesion and similarity between methods. Proceedings of the 44th annual Southeast regional conference, 2006: 91-95.[doi: 10.1145/1185448.1185469]

[21] Al Dallal J, Briand LC. A precise method-method interaction-based cohesion metric for object-oriented classes. ACM Transactions on Software Engineering and Methodology (TOSEM), 2012,21(2):1-34.[doi: 10.1145/2089116.2089118]

[22] Martin R. OO design quality metrics. An analysis of dependencies, 1994,12(1):151-170

[23] Li W, Henry S. Object-oriented metrics that predict maintainability. Journal of systems and software, 1993,23(2):111-122.[doi: 10.1016/0164-1212(93)90077-B]

[24] Briand L, Devanbu P, Melo W. An investigation into coupling measures for C++. Proceedings of the 19th international conference on Software engineering, 1997: 412-421.[doi: 10.1145/253228.253367]

[25] Briand LC, Daly JW, Wust JK. A unified framework for coupling measurement in object-oriented systems. IEEE Transactions on software Engineering, 1999,25(1):91-121.[doi: 10.1109/32.748920]

[26] Kim EM, Kusumoto S, Kikuno T, Chang OB. Heuristics for computing attribute values of C++ program complexity metrics. Proceedings of 20th International Computer Software and Applications Conference: COMPSAC'96, 1996: 104-109.[doi: 10.1109/CMPSAC.1996.542433]

[27] Basili VR, Rombach HD. The TAME project: Towards improvement-oriented software environments. IEEE Transactions on software engineering, 1988,14(6):758-773.[doi: 10.1109/32.6156]

[28] Harrison R, Counsell S, Nithi R. Coupling metrics for object-oriented design. Proceedings Fifth International Software Metrics Symposium. Metrics (Cat. No.98TB100262), 1998: 150-157.[10.1109/METRIC.1998.731240]

[29] Tang M-H, Kao M-H, Chen M-H. An empirical study on object-oriented metrics. Proceedings sixth international software metrics symposium (Cat. No. PR00403), 1999: 242-249.[doi: 10.1109/METRIC.1999.809745]

[30] Henderson-Sellers B Software metrics. Prentice Hall, Hemel Hempstaed, UK, 1996

[31] Tegarden DP, Sheetz SD, Monarchi DE. A software complexity model of object-oriented systems. Decision Support Systems, 1995,13(3-4):241-262.[doi: 10.1016/0167-9236(93)E0045-F]

[32] Lorenz M, Kidd J. Object‐oriented sofiware metrics. PTR Prentice Hall, Englewood Cliffs, New Jersey, U.S.A., 1994

[33] Lake A, Cook C. Use of factor analysis to develop OOP software complexity metrics. Proc. 6th Annual Oregon Workshop on Software Metrics, Silver Falls, Oregon, 1994.[doi: 10.1.1.56.7418]

[34] Benlarbi S, Melo WL. Polymorphism measures for early risk prediction. Proceedings of the 21st international conference on Software engineering, 1999: 334-344.[doi: 10.1109/ICSE.1999.841024]

[35] McCabe TJ. A complexity measure. IEEE Transactions on software Engineering, 1976,SE-2(4):308-320.[doi: 10.1109/TSE.1976.233837]

[36] Halstead MH. Elements of software science. Elsevier New York, 1977

[37] Curtis B, Sheppard SB, Milliman P, Borst M, Love T. Measuring the psychological complexity of software maintenance tasks with the Halstead and McCabe metrics. IEEE Transactions on software engineering, 1979,SE-5(2):96-104.[doi: 10.1109/TSE.1979.234165]

[38] Briand LC, Wüst J, Daly JW, Porter DV. Exploring the relationships between design measures and software quality in object-oriented systems. Journal of systems and software, 2000,51(3):245-273.[doi: 10.1016/S0164-1212(99)00102-8]

[39] Briand LC, Wust J. Modeling development effort in object-oriented systems using design properties. IEEE Transactions on Software Engineering, 2001,27(11):963-986.[doi: 10.1109/32.965338]

[40] Aggarwal K, Singh Y, Kaur A, Malhotra R. Software reuse metrics for object-oriented systems. Third ACIS Int'l Conference on Software Engineering Research, Management and Applications (SERA'05), 2005: 48-54.[doi: 10.1109/SERA.2005.60]

[41] Lima P, Guerra E, Meirelles P, Kanashiro L, Silva H, Silveira FF. A Metrics Suite for code annotation assessment. Journal of Systems and Software, 2018,137:163-183.[doi: 10.1016/j.jss.2017.11.024]