《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 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[1] |
| ***maxCC*** | 1976 | ***Maximum CC.*** CC的最大值. 即某个类方法中CC的最大值. | McCabe等[1] |
| ***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[1] |
| ***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[1] |
| ***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[2]和Curtis等[3] |
| ***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[4] |
| ***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[5] |
| ***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[5] |
| ***SIZE2*** | 1993 | *SIZE2*=一个类中属性数量+类中方法数量. | Li等[5] |
| ***NA*** | 2000 | ***Number of attributes.*** 某个类的属性数量, 包括继承和非继承. | Briand等[6] |
| ***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[7] |
| ***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[6] |
| ***NMIMP*** | 2001 | ***Number of methods implemented.*** 某个类实现的方法数量. | Briand等[7] |
| ***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[6] |
| ***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[6] |
| ***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[8] |
| ***CIS*** | 2002 | ***Class Interface Size.*** 类的接口规模. 与NPM相同. | Bansiya等[8] |
| ***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[8] |
| ***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[8] |
| ***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[9] |
| ***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[9] |
| ***SLOC*** | 1977 | ***Noncommentary source lines of code.*** 某个类中非注释的源代码行数. | Halstead等[2] |
| ***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[10] |
| ***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[10] |
| ***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[11] |
| ***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[11] |
| ***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[11] |
| ***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[11] |
| ***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[11] |
| ***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[11] |
| ***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[11] |

\*:与继承性NOP重名, 但定义不同.

**参考文献：**

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

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

[3] 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]

[4] 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]

[5] 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]

[6] 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]

[7] 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]

[8] 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]

[9] 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]

[10] 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]

[11] 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]