DEFINITIONS OF THE METRICS USED FOR MACHINE LEARNING

METRICS TAKEN FROM THE LITERATURE

Below, each *Metrics* is reported after other *Metrics* it depends on.

For each *Metric* the following information is reported:

- **Name**: the name of the *Metric;*
- **Definition**;
- **Computation Details:** all the important details about the computation, if needed;

Getter and setter are defined like this:

- with the name of "Accessor and Mutator" used by us for the sake of clarity (i.e., getter is an accessor, setter is a mutator),
- with the name "accessor" referred to either getter that setter, that is the terminology we often find in the literature.

LINES OF CODE (*LOC*)

Definition: the number of lines of code of an operation or a class, including blank lines and comments.

References: [53]

Computation Details:

- for method: we count the *LOC* from the method signature to the last curly bracket.
- For class: we count the *LOC* from the class declaration to the last curly bracket.
- For package: we sum the *LOC* of the classes declared in the package.
- For project: we sum the *LOC* of all packages.

LINES OF CODE WITHOUT ACCESSOR OR MUTATOR METHODS (LOCNAMM)

Definition: the number of lines of code of a class, including blank lines and comments and excluding accessor and mutator methods and corresponding comments.

References: [30]

Computation Details: we count the LOC from the class declaration to the last curly bracket.

NUMBER OF PACKAGES (*NOPK*)

Definition: total number of packages in a system.

References: -

NUMBER OF CLASSES (*NOCS*)

Definition: Total number of classes in a system, in a package or in a class.

References: -

Computation Details:

- for class: we sum up the number of nested classes.
- For package: we sum up the *NOCS* for all the classes in the package.
- For project: we sum up the *NOCS* for all the packages in the project.

NUMBER OF METHODS (*NOM*)

Definition: *NOM* represents the number of methods defined locally in a class, counting public as well as private methods. Overridden methods are not taken into account.

References: [29] [53]

Computation Details:

- for class: we sum up the number of *Methods Declared In Class*.
- For package: we sum up the *NOM* for all the classes in the package.
- For project: we sum up the *NOM* for all the packages in the project.

NUMBER OF NOT ACCESSOR OR MUTATOR METHODS (*NOMNAMM*)

Definition: *NOMNAMM* represents the number of methods defined locally in a class, counting public as well as private methods, excluding accessor or mutator methods.

References: [30]

Computation Details:

- for class: we sum up the number of Methods *Declared In Class* that are not accessor or mutator.
- For package: we sum up the *NOMNAMM* for all the classes in the package.
- For project: we sum up the *NOMNAMM* for all the packages in the project.

NUMBER OF ATTRIBUTES (NOA)

Definition: number of attributes of a class.

References: -

CYCLOMATIC COMPLEXITY (CYCLO)

Definition: *Cyclomatic Complexity* is the maximum number of linearly independent paths in a method. A path is linear if there is no branch in the execution flow of the corresponding code.

References: [29] [64]

Computation Details: we compute the strict cyclomatic complexity: with logical conjunction and logical and in conditional expressions also adding 1 to the complexity for each of their occurrences. i.e., the statement if (a && b || c) would have a cyclomatic complexity of one but a strict cyclomatic complexity of three. The minimum cyclomatic complexity is one.

WEIGHTED METHODS COUNT (WMC)

Definition: WMC is the sum of complexity of the methods that are defined in the class. We compute the complexity with the *Cyclomatic Complexity* metric (CYCLO).

References: [29] [17]

WEIGHTED METHODS COUNT OF NOT ACCESSOR OR MUTATOR METHODS (*WMCNAMM*)

Definition: *WMCNAMM* is the sum of complexity of the methods that are defined in the class, and are not accessor or mutator methods. We compute the complexity with the *Cyclomatic Complexity* metric (*CYCLO*).

References: [30]

AVERAGE METHODS WEIGHT (*AMW*)

Definition: the average static complexity of all methods in a class. We compute the complexity with the *Cyclomatic Complexity* metric (*CYCLO*).

Computation Details:

$$x = \begin{cases} AMW = \frac{WMC}{NOM}, & NOM \neq 0 \\ AMW = 0, & NOM = 0 \end{cases}$$

References: [61]

AVERAGE METHODS WEIGHT OF NOT ACCESSOR OR MUTATOR METHODS (*AMWNAMM*)

Definition: the average static complexity of all methods in a class, which are not accessor or mutator. We compute the complexity with the *Cyclomatic Complexity* metric (*CYCLO*).

Computation Details:

$$x = \begin{cases} AMWNAMM = \frac{WMCNAMM}{NOMNAMM}, & NOMNAMM \neq 0 \\ AMWNAMM = 0, & NOMNAMM = 0 \end{cases}$$

References: [29]

MAXIMUM NESTING LEVEL (MAXNESTING)

Definition: the maximum nesting level of control structures within an operation.

References: [49]

WEIGHT OF CLASS (WOC)

Definition: the number of "functional" public methods divided by the total number of public members.

References: [61]

Computation Details: we compute this metrics as:

$$\frac{Number\ of\ Non\ Abstract\ Public\ Non\ Accessor\ or\ Mutator\ Methods}{Total\ Number\ of\ Public\ Methods\ and\ Attribute}$$

If *Total Number of Public Methods and Attribute* is zero, WOC is zero.

CALLED LOCAL NOT ACCESSOR OR MUTATOR METHODS (CLNAMM)

Definition: the number of called not accessor or mutator methods declared in the same class of the measured method.

References: [30]

NUMBER OF PARAMETERS (NOP)

Definition: number of parameter of a method.

References: -

NUMBER OF ACCESSED VARIABLES (NOAV)

Definition: the total number of variables accessed directly or through accessor methods from the measured operation. Variables include parameters, local variables, but also instance variables and global variables declared in classes belonging to the system.

References: [49]

Computation Details: we count the *Used Variables* defined within the system and not in external libraries. The context we consider are: *MethodDeclarationParameter, CatchClause, EnumConstantDeclaration, VariableDeclarationExpression, VariableDeclarationStatement, SingleVariableDeclaration, VariableDeclarationFragment* [95]. To count the variables accessed through accessor methods we get the list of *Called Methods* and we count the *Used Intra Variables* by each accessor method in the set of *Called Methods*. We count also the variables accessed through static methods.

ACCESS TO LOCAL DATA (ATLD)

Definition: the number of attributes from the current classes accessed by the measured method directly or by invoking accessor methods.

References: [30]

Computation Details: we count the *Used Intra Variables* defined within the system and not in external libraries. To count the variable accessed through accessor methods we get the list of *Called Intra Methods* and we count the *Used Intra Variables* by each accessor method in the set of *Called Methods*.

NUMBER OF LOCAL VARIABLE (NOLV)

Definition: The total number of variables accessed directly or through accessor methods from the measured operation. Variables include parameters, local variables, but also instance variables and global variables declared in classes belonging to the system.

References: [61]

Computation Details: We count the *Used Variables* defined within the system and not in external libraries. The context we consider are: *MethodDeclarationParameter, CatchClause, EnumConstantDecla-*

ration, VariableDeclarationExpression, VariableDeclarationStatement, SingleVariableDeclaration, VariableDeclarationFragment [95]. To count the variable accessed through accessor methods we get the list of *Called Methods* and we count the *Used Intra Variables* by each accessor method in the set of *Called Methods*. We count also the variable access through static methods.

TIGHT CLASS COHESION (TCC)

Definition: *TCC* is the normalized ratio between the number of methods directly connected with other methods through an instance variable and the total number of possible connections between methods. A direct connection between two methods exists if both access the same instance variable directly or indirectly through a method call. *TCC* takes its value in the range [0, 1].

References: [29] [12] [13]

Computation Details: given:

Maximum number of possible connections: where N is the number of visible methods.

$$NP = \frac{N * (N - 1)}{2}$$

Number of direct connections: *NDC*, computed using a connectivity matrix that record all direct connected methods, making attention to cyclic calls among methods.

We compute:

$$TCC = \frac{NDC}{NP} \qquad NP \neq 0$$
$$TCC = 1 \qquad NP = 0$$

For *TCC* only visible methods are considered, i.e., they are not private or implement an interface or handle an event. Constructors are ignored. Constructors are a problem, because of indirect connections with attributes. They create indirect connections between methods which use different attributes, and increase cohesion, which is not real [29].

LACK OF COHESION IN METHODS (LCOM5)

Definition:

$$LCOM5 = \frac{NOM - \frac{_{m\epsilon M}\, NOAcc(m)}{NOA}}{NOM - 1}$$

where *M* is the set of methods of the class, *NOM* the number of methods, *NOA* the number of attributes, and *NOAcc*(m) is the number of attributes of the class accessed by method m

References: [29] [12]

Computation Details: For $_{m\epsilon M}\, NOAcc(m)$ we sum up *Used Intra Variables* by not constructor methods of the measured class.

Then we compute:

$$LCOM5 = \frac{NOM - \frac{\sum_{m \epsilon M} NOAcc(m)}{NOA}}{NOM - 1} \qquad NOM > 1 \wedge NOA > 0$$
$$LCOM5 = 0 \qquad NOM \leq 1 \vee \text{NOA} \leq 0$$

FANOUT

Definition: number of called classes.

References: [49]

Computation Details: we sum up the *Called Classes* belonging to the system.

ACCESS TO FOREIGN DATA (*ATFD*)

Definition: the number of attributes from unrelated classes belonging to the system, accessed directly or by invoking accessor methods.

References: [61]

Computation Details:

- For methods: we sum up the *Used Inter Variables* belonging to the system, also through not-*Constructor*, *Public* and not-*Abstract Called Inter Methods* of the system.
- For class: we sum up the *Used Inter Variable* belonging to the system, also through not *Constructor*, *Public* and not *Abstract Called Inter Methods* from the field declaration class of the methods and from all the not *Constructor* and not *Abstract Methods Declared In Class*. We do not declare a dependency to *ATFD* on method because we have to count each accessed variable only once.

FOREIGN DATA PROVIDERS (FDP)

Definition: the number of classes in which the attributes accessed - in conformity with the ATFD metric - are defined.

References: [49]

Computation Details: we sum up the classes where foreign data are defined, counting each class only once.

RESPONSE FOR A CLASS (RFC)

Definition: RFC is the size of the response set of a class. The response set of a class includes "all methods that can be invoked in response to a message to an object of the class". It includes local methods (also the inherited ones) as well as methods in other classes.

References: [29] [17]

Computation Details: we sum up also the Called Inter Methods and Called Hierarchy Methods by the Inherited Methods counting each method only one time. We consider only call to classes belonging to the system.

COUPLING BETWEEN OBJECTS CLASSES (CBO)

Definition: two classes are coupled if one of them uses the other, i.e., one class calls a method or accesses an attribute of the other class. Coupling involving inheritance and methods polymorphically called are taken into account. *CBO* for a class is the number of classes to which it is coupled.

References: [29] [17]

Computation Details: We sum up all the unrelated classes belonging to the system that define the variables and types used by the measured class and its *Ancestor Classes* and by methods the measured class methods declares and inherit. We count each class once.

CALLED FOREIGN NOT ACCESSOR OR MUTATOR METHODS (CFNAMM)

Definition:

- For method: the number of called not accessor or mutator methods declared in unrelated classes respect to the one that declares the measured method.
- For class: the number of called not accessor or mutator methods declared in unrelated classes respect to the measured one.

We consider only call to classes belonging to the system.

References: [30]

Computation Details: we sum up the number of not accessor or mutator *Called Inter Methods* and *Called Hierarchy Methods* of the system. We do not count the call to default constructor of classes.

COUPLING INTENSITY (CINT)

Definition: the number of distinct operations called by the measured operation.

References: [61]

Computation Details: we sum up *Called Inter Methods* belonging to system classes.

COUPLING DISPERSION (CDISP)

Definition: the number of classes in which the operations called from the measured operation are defined, divided by *CINT*.

References: [61]

Computation Details:

$$CDISP = \frac{FANOUT}{CINT} \qquad CINT \neq 0$$
$$CDISP = 0 \qquad CINT = 0$$

MAXIMUM MESSAGE CHAIN LENGTH (MAMCL)

Definition: the maximum length of chained calls in a method.

References: [30]

Computation Details: we compute the maximum length of all call chains in a method. Call chains have a minimum length of two.

NUMBER OF MESSAGE CHAIN STATEMENTS (NMCS)

Definition: the number of different chained calls in a method.

References: [30]

Computation Details: we compute the number of call chains in a method. Call chains have a minimum length of two.

MEAN MESSAGE CHAIN LENGTH (MEMCL)

Definition: the average length of chained calls in a method.

References: [30]

Computation Details: we compute the rounded average length of all call chains in a method. Call chains have a minimum length of two. If *NMCS* is zero, then *MeMCL* is zero too.

CHANGING CLASSES (CC)

Definition: the number of classes in which the methods that call the measured method are defined in.

References: [61]

CHANGING METHODS (CM)

Definition: the number of distinct methods that call the measured method.

References: [61]

NUMBER OF ACCESSOR METHODS (NOAM)

Definition: the number of accessor (getter and setter) methods of a class.

References: [49]

NUMBER OF PUBLIC ATTRIBUTES (NOPA)

Definition: the number of public attributes of a class.

References: [49]

LOCALITY OF ATTRIBUTE ACCESSES (LAA)

Definition: the number of attributes from the method's definition class, divided by the total number of variables accessed (including attributes used via accessor methods, see *ATFD*), whereby the number of local attributes accessed is computed in conformity with the *ATLD* specifications. We consider only variables declared in system classes.

References: [49]

Computation Details: each attribute count only one, independently from the ways the class accesses it (e.g., directly or through accessor and/or mutator) and how many times accesses it.

DEPTH OF INHERITANCE TREE (DIT)

Definition: the depth of a class, measured by *DIT*, within the inheritance hierarchy is the maximum length from the class node to the root of the tree, measured by the number of ancestor classes. *DIT* has a minimum value of one, for classes that do not have ancestors.

References: [29] [17]

Computation Details: we consider only hierarchy classes belonging to the system: we visit the *Ancestor Classes* in a bottom up order and we stop counter at the first class that does not belong to the system.

NUMBER OF INTERFACES (NOI)

Definition: number of interfaces declared in a package or in a system.

References: -

NUMBER OF CHILDREN (NOC)

Definition: number of children counts the immediate subclasses subordinated to a class in the class hierarchy.

References: [29] [17]

NUMBER OF METHODS OVERRIDDEN (NMO)

Definition: *NMO* represents the number of methods that have been overridden i.e., defined in the superclass and redefined in the class. This metric includes methods doing super invocation to their parent method. *NMO* is not defined for classes that have not superclass.

References: [29] [53]

NUMBER OF INHERITED METHODS (NIM)

Definition: *NIM* is a simple measure showing the amount of behaviour that a given class can reuse. It counts the number of methods that a class can access in its superclasses. *NIM* is not defined for classes that have not superclass.

References: [29] [53]

NUMBER OF IMPLEMENTED INTERFACES (NOII)

Definition: number of implemented interfaces by a class.

References: -

CUSTOM METRICS DEFINED FOR MACHINE LEARNING

For Each *Custom Metrics* the following information is reported:

- **Name**: the name of *Custom Metrics;*
- **Definition**.

NUMBER OF DEFAULT ATTRIBUTES (NODA)

Definition: number of attributes with default visibility by a class. The default visibility is equal to a non – protected, non – private and non – public visibility.

NUMBER OF PRIVATE ATTRIBUTES (NOPVA)

Definition: number of private attributes by a class.

NUMBER OF PROTECTED ATTRIBUTES (NOPRA)

Definition: number of protected attributes by a class.

NUMBER OF FINAL ATTRIBUTES (NOFA)

Definition: number of final attributes by a class.

NUMBER OF FINAL AND STATIC ATTRIBUTES (NOFSA)

Definition: number of final and static attributes by a class.

NUMBER OF FINAL AND NON - STATIC ATTRIBUTES (NOFNSA)

Definition: number of final and non - static attributes by a class.

NUMBER OF NOT FINAL AND NON - STATIC ATTRIBUTES (NONFNSA)

Definition: number of non - final and non - static attributes by a class.

NUMBER OF STATIC ATTRIBUTES (NOSA)

Definition: number of static attributes by a class.

NUMBER OF NON - FINAL AND STATIC ATTRIBUTES (NONFSA)

Definition: number of non – final and static attributes by a class.

NUMBER OF ABSTRACT METHODS (NOABM)

Definition: number of abstract methods by a class.

NUMBER OF CONSTRUCTOR METHODS (NOCM)

Definition: number of constructor methods by a class.

NUMBER OF NON - CONSTRUCTOR METHODS (NONCM)

Definition: number of non - constructor methods by a class.

NUMBER OF FINAL METHODS (NOFM)

Definition: number of final methods by a class.

NUMBER OF FINAL AND NON - STATIC METHODS (NOFNSM)

Definition: number of final and non – static methods by a class.

NUMBER OF FINAL AND STATIC METHODS (NOFSM)

Definition: number of final and static methods by a class.

NUMBER OF NON - FINAL AND NON - ABSTRACT METHODS (NONFNABM)

Definition: number of non - final and non - abstract methods by a class.

NUMBER OF FINAL AND NON - STATIC METHODS (NONFNSM)

Definition: number of final and non - static methods by a class.

NUMBER OF NON - FINAL AND STATIC METHODS (NONFSM)

Definition: number of non - final and static methods by a class.

NUMBER OF NON - FINAL AND STATIC METHODS (NONFSM)

Definition: number of non - final and static methods by a class.

NUMBER OF DEFAULT METHODS (NODM)

Definition: number of methods with default visibility by a class. The default visibility is equal to a non – protected, non – private and non – public visibility.

NUMBER OF PRIVATE METHODS (NOPM)

Definition: number of methods with private visibility by a class.

NUMBER OF PROTECTED METHODS (NOPRM)

Definition: number of protected methods by a class.

NUMBER OF PUBLIC METHODS (NOPLM)

Definition: number of public methods by a class.

NUMBER OF NON - ACCESSORS METHODS (NONAM)

Definition: number of non – accessors methods by a class.

NUMBER OF STATIC METHODS (NOSM)

Definition: number of static methods by a class.

REFERENCES

[12] L. Briand, J. Daly, and J. Wust, "A unified framework for coupling measurement in object-oriented systems," *IEEE Transactions on Software Engineering*, vol. 25, no. 1, pp. 91–121, 1999.

[13] W. J. Brown, R. C. Malveau, T. J. Mowbray, and J. Wiley, *AntiPatterns - Refactoring Software , Architectures , and Projects in Crisis*. T. Hudson, Ed. Robert Ipsen, 1998.

[17] O. Ciupke, "Automatic detection of design problems in object-oriented reengineering," in *Proceedings of the Technology of Object-Oriented Languages and Systems (TOOLS 1999)*, 1999, Santa Barbara, California: IEEE Comput. Soc, pp. 18–32. doi:10.1109/TOOLS.1999.787532.

[29] E. Van Emden, E. van Emden, and L. Moonen, "Java Quality Assurance by Detecting Code Smells," in *Proceedings of the Ninth Working Conference on Reverse Engineering (WCRE 2002)*, 2002, Richmond, USA: IEEE, pp. 97–106. doi:10.1109/WCRE.2002.1173068.

[30] V. Ferme, "JCodeOdor: A Software Quality Advisor Through Design Flaws Detection," Università degli Studi di Milano-Bicocca, 2013.

[49] M. Lanza and R. Marinescu, *Object-Oriented Metrics in Practice*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, p. 212. doi:10.1007/3-540-39538-5.

[53] M. Lorenz and J. Kidd, *Object-oriented software metrics: a practical guide*. Prentice-Hall, Ed. 1994.

[61] R. Marinescu, "Measurement and quality in object-oriented design," Politehnica University of Timisoara, 2002.

[64] T. McCabe, "A complexity measure," *IEEE Transactions on Software Engineering*, vol. SE-2, no. 4, pp. 308–320, 1976. doi:10.1109/TSE.1976.233837.

[95] "Eclipse JDT Programmer's Guide." [Online]. Available: http://help.eclipse.org/kepler/index.jsp?topic=%2Forg.eclipse.jdt.doc.isv%2Fguide%2Fjdt_int.htm.