Metrics Definitions

All the following Halstead metrics share the following definitions:

- *n1*: number of distinct operators (semantic meanings of the reserved keywords, semicolons, blocks, and identifiers except in their declarations)
- *n2*: number of distinct operands (literals - e.g. character, string, and integer literals, - and the identifiers in their declarations)
- *N1*: total number of operators
- *N2*: total number of operands
- *n : n1 + n2* (program vocabulary)
- *N : N1 + N2* (program length)
- *V : N * log2(n)* (volume)
- *D : n1/2 * N2/n2* (difficulty)
- *E : D * V (effort)*

Halstead Calculated Program Length (HCPL)

The calculated program length is $HCPL = n1 * log_2(n1) + n2 * log_2(n2)$.

Halstead Difficulty (HDIF)

The Halstead difficulty is $HDIF = n1/2 * N2/n2$.

Halstead Effort (HEFF)

The Halstead effort is $HEFF = D * V$.

Halstead Number of Delivered Bugs (HNDB)

Number of delivered bugs is $HNDB = E^{2/3}/3000$.

Halstead Program Length (HPL)

Halstead program length is $HPL = N1 + N2$.

Halstead Program Vocabulary (HPV)

Halstead program vocabulary is $HPV = n1 + n2$.

Halstead Time Required to Program (HTRP)

Halstead time required to program is $HTRP = E/18$ seconds.

Halstead Volume (HVOL)

The Halstead volume is $HVOL = N * log_2(n)$.

Maintainability Index (Microsoft version) (MIMS)

The Maintainability Index used by Microsoft's Visual Studio is computed by the following formula:

$MIMS = max(0,(171 - 5.2 * ln(HVOL) - 0.23 * (McCC) - 16.2 * ln(LLOC)) * 100 / 171)$

Maintainability Index (Original version) (MI)

The original Maintainability Index is computed by the following formula:

*MI = 171 - 5.2 * ln(HVOL) - 0.23 * (McCC) - 16.2 * ln(LLOC)*

Maintainability Index (SEI version) (MISEI)

The Maintainability Index derived by the Software Engineering Institute (SEI) is computed by the following formula:

*MISEI = 171 - 5.2 * log_2(HVOL) - 0.23 * McCC - 16.2 * log_2(LLOC) + 50 * sin(sqrt(2.4 * CD))*

Maintainability Index (SourceMeter version) (MISM)

The Maintainability Index proposed by SourceMeter combines the different scaling approach from Microsoft's version with the inclusion of comment percentage from the Software Engineering Institute (SEI) version into the following formula:

*MISM = max(0, (171 - 5.2 * log_2(HVOL) - 0.23 * McCC - 16.2 * log_2(LLOC) + 50 * sin(sqrt(2.4 * CD))) * 100 / 171)*

McCabe's Cyclomatic Complexity (McCC)

Complexity of the method expressed as the number of independent control flow paths in it. It represents a lower bound for the number of possible execution paths in the source code and at the same time it is an upper bound for the minimum number of test cases needed for achieving full branch test coverage. The value of the metric is calculated as the number of the following instructions plus 1: if, for, foreach, while, do-while, case label (which belongs to a switch instruction), catch, conditional statement (?:). Moreover, logical "and" (&&) and logical "or" (||) expressions also add 1 to the value because their short-circuit evaluation can cause branching depending on the first operand. The following instructions are not included: else, switch, default label (which belongs to a switch instruction), try, finally.

Nesting Level (NL)

Complexity of the method expressed as the depth of the maximum embeddedness of its conditional, iteration and exception handling block scopes. The following instructions are taken into account: if, else-if, else, for, while, do-while, switch, try, catch, finally and block statements that are directly inside another block statement. The following instructions do not increase the value by themselves; however, if additional embeddednesses can be found in their blocks, they are considered: case and default label (which belong to a switch instruction).

Nesting Level Else-If (NLE)

Complexity of the method expressed as the depth of the maximum embeddedness of its conditional, iteration and exception handling block scopes, where in the if-else-if construct only the first if instruction is considered. The following instructions are taken into account: if, else, for, while, do-while, switch, try, catch, finally and block statements that are directly inside another block statement. The following instructions do not increase the value by themselves; however, if additional embeddednesses can be found in their blocks, they are considered: else-if (i.e. in the if-else-if construct the use of else-if does not increase the value of the metric), case and default label (which belong to a switch instruction).

Number of Incoming Invocations (NII)

Number of other methods and attribute initializations which directly call the method. If the method is invoked several times from the same method or attribute initialization, it is counted only once.

Number of Outgoing Invocations (NOI)

Number of directly called methods. If a method is invoked several times, it is counted only once.

Comment Density (CD)

Ratio of the comment lines of the method (CLOC) to the sum of its comment (CLOC) and logical lines of code (LLOC).

Comment Lines of Code (CLOC)

Number of comment and documentation code lines of the method; however, its anonymous and local classes are not included.

Documentation Lines of Code (DLOC)

Number of documentation code lines of the method.

Total Comment Density (TCD)

Ratio of the total comment lines of the method (TCLOC) to the sum of its total comment (TCLOC) and total logical lines of code (TLLOC).

Total Comment Lines of Code (TCLOC)

Number of comment and documentation code lines of the method, including its anonymous and local classes.

Lines of Code (LOC)

Number of code lines of the method, including empty and comment lines; however, its anonymous and local classes are not included.

Logical Lines of Code (LLOC)

Number of non-empty and non-comment code lines of the method; however, its anonymous and local classes are not included.

Number of Parameters (NUMPAR)

Number of the parameters of the method. The varargs parameter counts as one.

Number of Statements (NOS)

Number of statements in the method; however, the statements of its anonymous and local classes are not included.

Clone Classes (CCL)

Number of clone classes having at least one clone instance in the source code element.

Clone Complexity (CCO)

Sum of CCO of clone instances in the source code element.

Clone Coverage (CC)

Ratio of code covered by code duplications in the source code element to the size of the source code element, expressed in terms of the number of syntactic entities (statements, expressions, etc.).

Clone Instances (CI)

Number of clone instances in the source code element.

Clone Line Coverage (CLC)

Ratio of code covered by code duplications in the source code element to the size of the source code element, expressed in terms of lines of code.

Clone Logical Line Coverage (CLLC)

Ratio of code covered by code duplications in the source code element to the size of source code element, expressed in terms of logical lines of code (non-empty, non-comment lines).

Lines of Duplicated Code (LDC)

Number of code lines covered by code duplications in the source code element.

Logical Lines of Duplicated Code (LLDC)

Number of logical code lines (non-empty, non-comment lines) covered by code duplications in the source code element.