*Clear*SQL Code Metrics Inside

THE CODE METRICS FEATURE DETERMINES THE COMPLEXITY OF CODE AND HIGHLIGHTS COMPLEX AND HARD TO MAINTAIN CODE BY FLAGGED METRICS. IT IDENTIFIES POTENTIAL PROBLEM AREAS BASED ON COMPLEXITY, SIZE AND MODULARITY.

IT INDICATES IT WITH A RED FLAG WHEN THE METRIC VALUE IS HIGHER OR LOWER THAN ITS DEFINITION IN THE CODE METRICS OPTIONS.

*Clear*SQL calculates four main code metrics:

- CYCLOMATIC COMPLEXITY
- MAINTAINABILITY INDEX
- HALSTEAD SOFTWARE METRICS
- INTERFACE COMPLEXITY

CYCLOMATIC COMPLEXITY

McCabe's Cyclomatic complexity is a software metric. It was developed and described by Thomas J. McCabe, Sr. in 1976 in the 1975 SE-2(4) "IEEE Transactions on Software Engineering", and is used to indicate the complexity of a program. The value of this metric is the number of distinct paths through the code in a program. If a program has no conditional statements, the cyclomatic complexity will be one.

Cyclomatic complexity is computed using the control flow graph of the program: the nodes of the graph correspond to indivisible groups of commands of a program, and a directed edge connects two nodes if the second command might be executed immediately after the first command.

A value greater than 10 indicates that the routine is probably hard to identify and test. You can define the maximum routine Cyclomatic Complexity in the Code Analyzer Options in the Code Metrics Options page.

Average cyclomatic complexity of subprograms

Vg_avg_of_subprogams* = Sum(*Vg*) / *SubN

Vg – Cyclomatic complexity of a subprogram
SubN – Number of subprograms

Cyclomatic complexity of a objects

Vg_of_object* = Sum(*Vg*) + *Vg_init

Vg – Cyclomatic complexity of a subprogram
Vg_init - Cyclomatic complexity of the initialization part of a package

Average cyclomatic complexity of objects

Vg_avg_of_objects* = Sum(*Vg_of_object*) / *ObjN

Vg_of_object - Cyclomatic complexity of a object
ObjN – Number of objects

Cyclomatic complexity of a script

Vg_of_script = Sum(Vg_of_object) + Vg_of_rest_code

Vg_of_object - Cyclomatic complexity of the object in a script
Vg_of_rest_code - Cyclomatic complexity of all rest code (code that do not belong to any object)

Average cyclomatic complexity of scripts

Vg_avg_of_scripts** = Sum(**Vg_of_script**) / **ScriptN

Vg_of_script - Cyclomatic complexity of a script
ScriptN – Number of scripts

Total cyclomatic complexity of a *Clear*SQL-Project

... is the sum of Cyclomatic Compexity metric scores of all scripts in a *Clear*SQL project.

MAINTAINABILITY INDEX

In 1992 Oman proposed the following polynomial expression to determine the Maintainability Index (MI) for a program. Maintainability Index (MI) is a composite metric that incorporates a number of traditional source code metrics into a single number that indicates relative maintainability. It measures how maintainable (easy to support and change) the source code is. The Maintainability is based on the Halstead Volume (HV) metric, the Cyclomatic Complexity (CC) metric, the average number of lines of code per module (LOC), and the percentage of comment lines per module (COM). The higher the MI, the more maintainable a system is deemed to be.

Maintainability index of a subprogram

MI = 171 -5.2 * ln(HV) - 0.23 * CC - 16.2 * ln (LOC) + 50 * sin(sqrt(2.4 * COM))

HV –Halstead Volume of a subprogram

CC - Cyclomatic Complexity of a subprogram

LOC - number of lines of code of a subprogram

COM - proportion/percentage of comment lines of a subprogram
This parameter depends of the *model* selected in the "Code Analyzer Options" of ***Clear*SQL**:

- **"Model 1"** (=default), available in *Clear*SQL since version 6.5.7.126, interprets COM as percentage of lines of comments, ranging from 1 to 100. Interpreting COM as a percentage leads to an influence that is alternating between positive and negative.
- **"Model 2"**, interprets COM as a proportion of lines of commenst, ranging from 0 to 1. Using this interpretation results in a monotonously increasing value for MI. This relationship means as higher the proportion of comments in the code, the better.

Average maintainability index of subprograms

MI_avg_of_subprograms = Sum(MI) / SubN

MI – Maintainability index of a subprogram
SubN – Number of subprograms

Maintainability index of a object

MI_of_object = 171 -5.2 * ln(avg_o_HV) - 0.23 * avg_o_CC - 16.2 * ln (avg_o_LOC) + 50 * sin(sqrt(2.4 * avg_o_COM))

avg_o_HV – average Halstead Volume of “object modules”. “Object module” is a standalone subprogram by itself, a object subprogram or a initialization part of a package.

avg_o_CC - average Cyclomatic Complexity of “object modules”.

avg_o_LOC - average number of lines of code of “object modules”.

avg_o_COM - average proportion/percentage of comment lines of “object modules”. This parameter depends of the *model* selected in the "Code Analyzer Options" of ***Clear*SQL**
(see "Maintainability index of a subprogram").

Average maintainability index of objects

MI_avg_of_objects = *Sum*(***MI_of_object***) / ***ObjN***

MI_of_object – Maintainability index of a object
ObjN – Number of objects

Maintainability index of a script

MI_of_script *= 171 -5.2 * ln(**avg_s_HV**) - 0.23 * **avg_s_CC** - 16.2 * ln (**avg_s_LOC**) + 50 * sin(sqrt(2.4 * **avg_s_COM**))*

avg_s_HV – average Halstead Volume of “script modules”. “Script module” is a standalone subprogram, a object subprogram, a initialization part of a package or an anonymous block.
avg_s_CC - average Cyclomatic Complexity of “script modules”.
avg_s_LOC - average number of lines of code of “script modules”.
avg_s_COM - average proportion/percentage of comment lines of “script modules”. This parameter depends on the *model* selected in the "Code Analyzer Options" of ***Clear*SQL**.
(see "Maintainability index of a subprogram").

Average maintainability index of scripts

MI_avg_of_scripts *= Sum(**MI_of_script**) / **ScriptN***

MI_of_script – Maintainability index of a script
ScriptN – Number of scripts

Total maintainability index

MI_total = 171 -5.2 * ln(avg_m_HV) - 0.23 * avg_m_CC - 16.2 * ln (avg_m_LOC) + 50 * sin(sqrt(2.4 * avg_m_COM))

avg_m_HV – average Halstead Volume of “modules”. “Module” is a standalone subprogram, a object subprogram, a initialization part of the package, a anonymous PL/SQL block or a separate SQL statement.
avg_m_CC - average Cyclomatic Complexity of “modules”.
avg_m_LOC - average number of lines of code of “modules”.
avg_m_COM - average proportion/percentage of comment lines of “modules”. This parameter depends on the *model* selected in the "Code Analyzer Options" of ***Clear*SQL**.
(see "Maintainability index of the subprogram")

HALSTEAD SOFTWARE METRICS

Halstead software metrics were proposed by Maurice Howard Halstead in 1977. It is a group of metrics related to computational complexity.

The metrics are derived from four low-level metrics obtained directly from the source code.

n1 - The number of distinct operators in a subprogram
n2 - The number of distinct operands in a subprogram
N1 - The total number of operators in a subprogram
N2 - The total number of operands in a subprogram

The following five metrics of the subprogram are calculated from these four primitives:

1. **Program Length**
   N = N1+ N2
2. **Program Vocabulary**
   n = n1+ n2
3. **Volume**
   V = N * log2(n)
4. **Difficulty**
   D = n1/2 * (N2/n2)
5. **Effort**
   E = D * V

Average Halstead Volume of subprograms

V_avg_of_subprogams = *Sum(**V**) / **SubN***

V – Halstead Volume of a subprogram
SubN – Number of subprograms

Halstead Volume of a object

V_of_object = *Sum(**V**)* + ***V_init***

V – Halstead Volume of a subprogram
V_init - Halstead Volume of the initialization part of a package

Average Halstead Volume of objects

V_avg_of_objects = *Sum(**V_of_object**) / **ObjN***

V_of_object - Halstead Volume of the object
ObjN – Number of objects

Halstead Volume of a script

V_of_script = Sum(V_of_object) + V_of_rest_code

V_of_object - Halstead Volume of a object in a script
V_of_rest_code - Halstead Volume of all rest code *(code that do not belong to any object)*

Average Halstead Volume of scripts

V_avg_of_scripts = *Sum(**V_of_script**) / **ScriptN***

V_of_script - Halstead Volume of a script
ScriptN – Number of scripts

Total Halstead Volume of a *Clear*SQL-Project

... is the sum of Halstead Volume metric of all scripts in *Clear*SQL-Project

INTERFACE COMPLEXITY

Interface complexity of a subprogram

InterfaceComplexity = r_params + ret_p

r_params - Required input parameters of a subprogram (excluding the parameters with default values)
ret_p - Subprogram return points

Average Interface Complexity of subprograms, Interface Complexity of a object, Average Interface Complexity of objects, Interface Complexity of a script, Average Interface Complexity of scripts and Total Interface Complexity of a *Clear*SQL-Project are calculated in a similar manner as the corresponding metrics based on Cyclomatic Complecity or Halstead Volume.

FUNCTIONAL COMPLEXITY

Functional complexity of a subprogram

FunctionalComplexity = InterfaceComplexity + V

InterfaceComplexity - Interface Complexity of a subprogram
V - Halstead Volume of a subprogram

Other minor metrics derived directly from a source code

LOC - number of lines of code of a subprogram. Calculated as number of all lines in the subprogram excluding the blank lines and the comment lines.

eLOC - number of effective lines of code of a subprogram. It is the same as LOC excluding BEGIN and END lines of a PL/SQL block, END IF and END LOOP lines.

lsLOC - number of logical statements of a subprogram. Calculated as the number of effective lines that have semicolon as a trailing character.