Software Engineering | Halstead's Software Metrics

A computer program is an implementation of **an algorithm** considered to be a collection of **tokens** which can be classified as either **operators or operands**.

Halstead's metrics are included in a number of current commercial tools that **count software lines of code**. By **counting the tokens and determining which are operators and which are operands**,

the following base measures can be collected :

n1 = Number of distinct operators.
n2 = Number of distinct operands.
N1 = Total number of occurrences of operators.
N2 = Total number of occurrences of operands.

In addition to the above, Halstead defines the following :

n1* = Minimum Possible Number of potential operators.
n2* = Minimum Possible Number of potential operands.

Halstead refers to n1* and n2* as the **minimum possible number of operators and operands for a module and a program respectively**. This minimum number would be embodied in the programming language itself, in which the required operation would already exist

(for example, in C language, any program must contain at least the definition of the function main()), possibly as a function or as a procedure:

n1* = 2, since at least 2 operators must appear for any function or procedure :

1 for the name of the function and 1 to serve as an assignment or grouping symbol, and n2* represents the number of parameters, without repetition, which would need to be passed on to the function or the procedure.

Halstead metrics –

Halstead metrics are :

- **Halstead Program Length –** The total number of operator occurrences and the total number of operand occurrences.

  $N = N1 + N2$

- **Halstead Vocabulary –** The total number of unique operator and unique operand occurrences.

  $n = n1 + n2$

- **Program Volume (V) –** Proportional to program size, represents the size, in bits, of space necessary for storing the program. This parameter is dependent on specific algorithm implementation.

- **Potential Minimum Volume –** The potential minimum volume V* is defined as the volume of the program in which **a problem can be coded**.

  $V^* = (2 + n2^*) * \log_2(2 + n2^*)$

  Here, n2* is the count of unique input and output parameters

- **Program Level –** To rank the programming languages, the level of abstraction provided by the programming language, Program Level (L) is considered. The higher the level of a language, the less effort it takes to develop a program using that language.

  $L = V^* / V$

  The value of L ranges between zero and one, with L=1 representing a program written at the highest possible level (i.e., with minimum size).

- **Program Difficulty –** This parameter shows how difficult to handle the program is.
  $D = (n1 / 2) * (N2 / n2)$
  $D = 1 / L$

- **Programming Effort –** Measures the amount of **mental activity** needed to translate the existing algorithm into implementation in the specified program language.
  $E = V / L = D * V$

  E= Difficulty * Volume

- **Language Level –** Shows the algorithm implementation program language level. The same algorithm demands additional effort if it is written in a low-level program language. **For example, it is easier to program in Pascal than in Assembler**.

  $$L' = V / D / D$$
  $$\text{lambda} = L * V* = L^{2} * V$$

- **Intelligence Content –** Determines the amount of **intelligence** presented (stated) in the program This parameter provides a measurement of program complexity, independently of the program language in which it was implemented.

  $$I = V / D$$

- **Programming Time –** Shows time (in minutes) needed to translate the existing algorithm into implementation in the specified program language.

  $$T = E / (f * S)$$

  5 <= S <= 20. Halstead uses 18. The value of S has been empirically developed from psychological reasoning, and its recommended value for programming applications is 18.

  number S = 18 moments / second

  seconds-to-minutes factor f = 60