

# An Analysis of the Design and Definitions of Halstead's Metrics

By

Rafa AL-QUTAISH & Alain ABRAN

---

The 15th International Workshop on Software Measurement  
(IWSM'2005)

September 12-14, 2005  
Montréal (Québec) CANADA

# An Analysis of the Design and Definitions of Halstead's Metrics

---

## AGENDA

- ➔ Introduction
- ➔ The Analysis Framework
- ➔ Halstead's Metrics
- ➔ Analysis of Halstead's Metrics
- ➔ Discussion and Conclusions
- ➔ Questions?

# Introduction

---

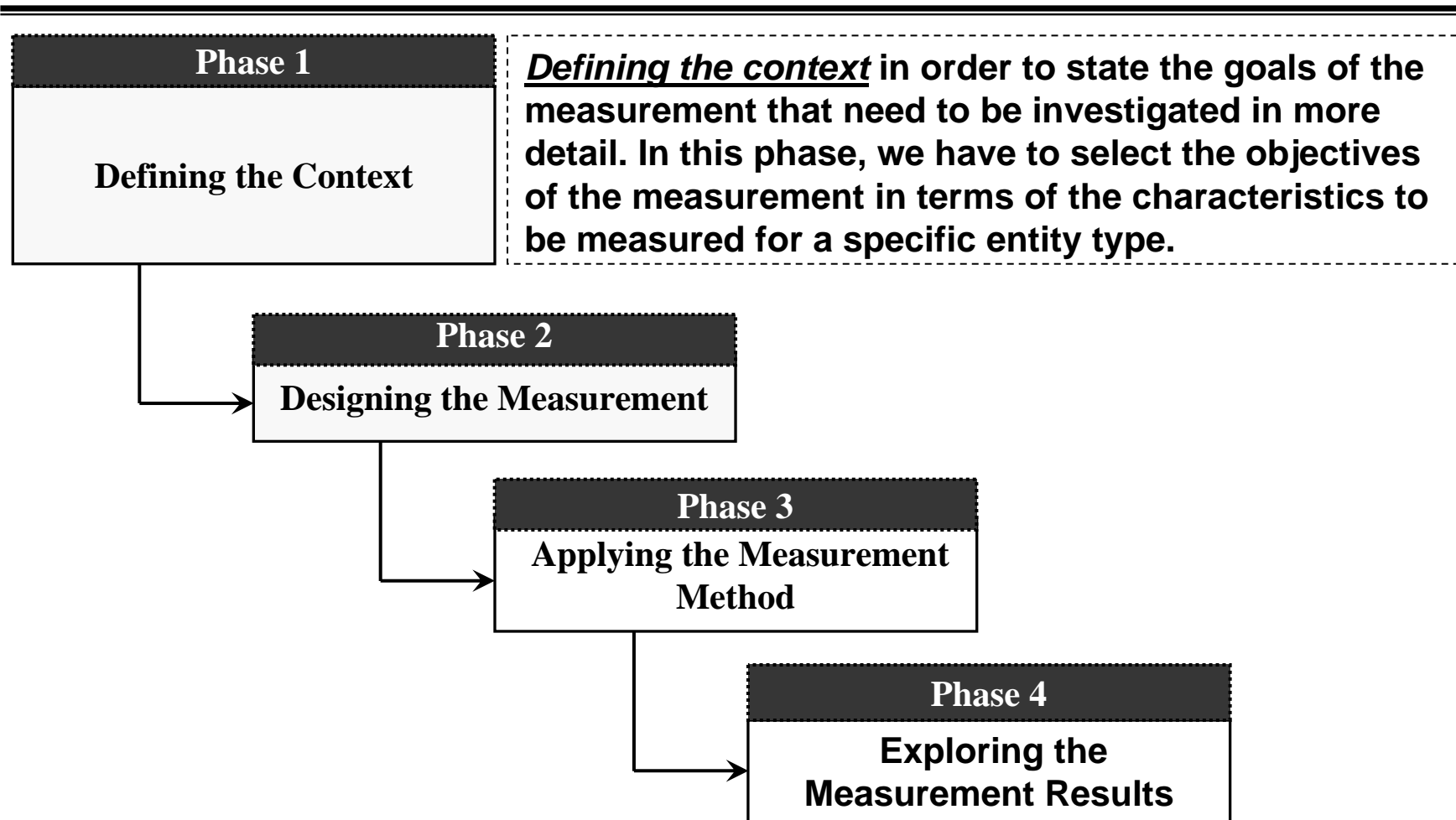
- In 1977, Halstead introduced his set of Software Metrics which are commonly referred to as ‘software science’.
- Halstead’s Metrics were used by researchers to:
  - Measure open source software,
  - Incorporate software measurement into a compiler,
  - Measure programs written in functional programming languages,
  - Measure software written for a real-time switching system,
  - Evaluate student programs and query languages, . . . etc.
- In this paper we will investigate the various elements of the design and definitions of Halstead’s metrics based on a Software Measurement Analysis Framework proposed by Habra *and others* in 2004.
- This Analysis Framework was used in [Abran *and others*, 2004] to analyse McCabe Cyclomatic Complexity Number.

# The Analysis Framework

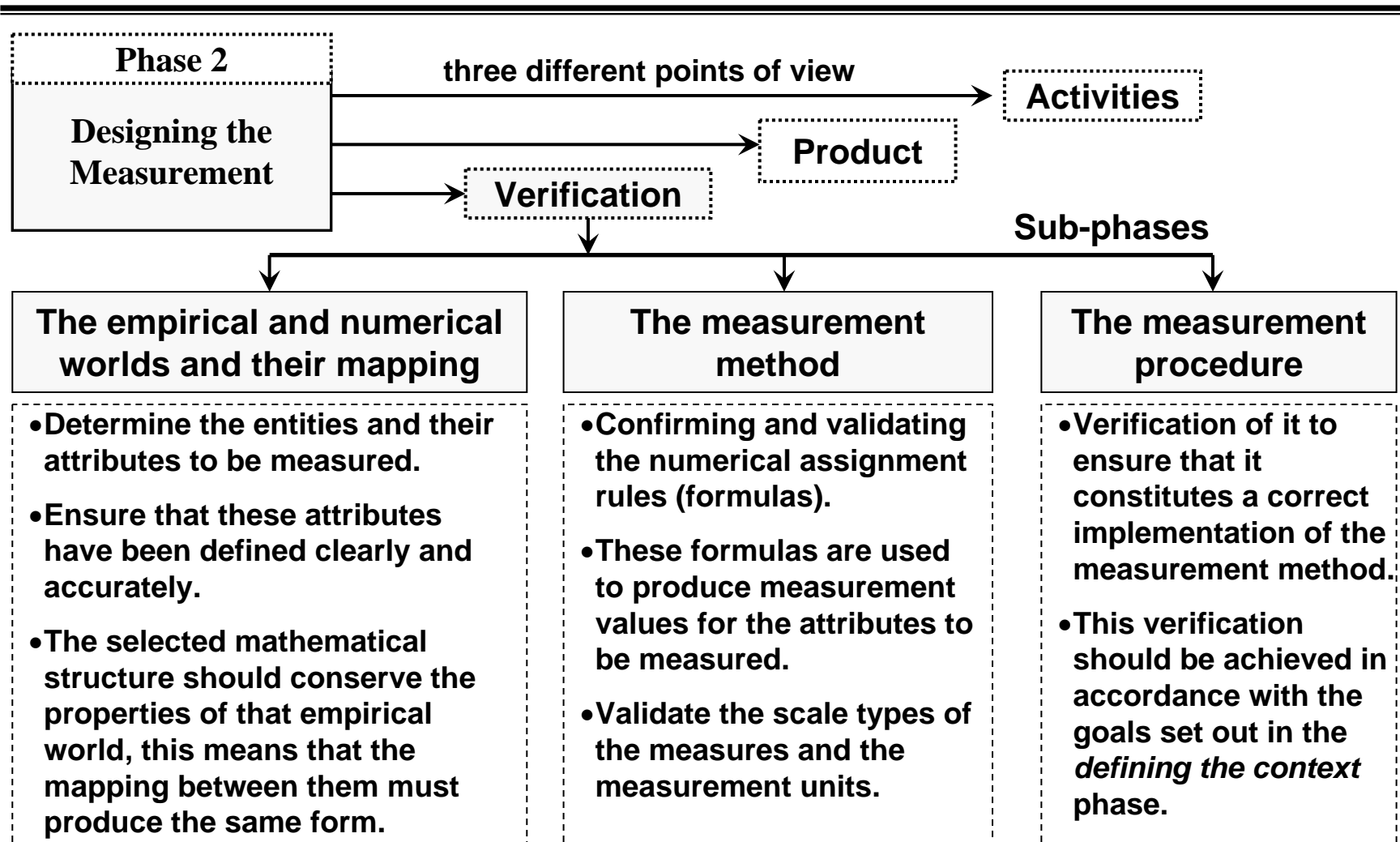
---

- It is based on a work by Jacquet and Abran in 1997.
- It consists of four phases of the software measurement life cycle:
  - *defining the context*,
  - designing the measurement,
  - applying the measurement method, and
  - exploring the measurement results.
- Can be used to investigate and verify existing software measures.
- To analyze the design and definitions of Halstead's metrics, we need to apply the first two phases of this analysis framework:
  - *Defining the context*, and
  - *Designing the measurement*.
- Next we will discuss these the first two phases in more details.

# The Analysis Framework (cont.)



# The Analysis Framework (cont.)



# Halstead's Metrics

---

- According to Halstead, a computer program is an implementation of an algorithm considered to be a collection of tokens that can be classified as either operators or operands.
- By counting the tokens and determining which are operators and which are operands based on a counting strategy, 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:
  - n1\*: Number of potential operators (the minimum possible number of operators for a module or a program).
  - n2 \*: Number of potential operands (the minimum possible number of operands for a module or a program).

## Halstead's Metrics (cont.)

---

- All of the Halstead's metrics are defined based on the above collective measures ( $n_1$ ,  $n_2$ ,  $N_1$ ,  $N_2$ ,  $n_1^*$  and  $n_2^*$ ), Halstead defines the following metrics (derived measures):

- The *length* ( $N$ ) of a program  $P$  is:

$$N = N_1 + N_2. \quad (1)$$

- The *vocabulary* ( $n$ ) of a program  $P$  is:

$$n = n_1 + n_2. \quad (2)$$

- *Program volume* ( $V$ ) is:

$$V = N * \log_2 n. \quad (3)$$

Program volume ( $V$ ) is defined by Halstead in his book as:

- 1) a suitable metric for the size of any implementation of any algorithm [Halstead's book, p. 19].
- 2) a count of the number of mental comparisons required to generate a program [Halstead's book, p. 47].

Length, Vocabulary, and Volume → Different views of program size.



## Halstead's Metrics (cont.)

---

- Program *potential (minimal) volume* ( $V^*$ ), , which is the volume of the minimal size implementation of a program P, is defined as:

$$V^* = (2 + n_2) \log_2 (2 + n_2). \quad (4)$$

No objective evidence documented in Halstead's book that this is indeed a minimal implementation.

- Program level (L) of a program P with volume V is:

$$L = \frac{V^*}{V}. \quad (5)$$

- Growth in volume leads to a lower level of program.
- This value is interpreted as referring to the most ideally written program and as measuring how well written a program is.
- Programs with L values close to 1 are considered to be well written, in general  $L < 1$ .

## Halstead's Metrics (cont.)

---

- Program *difficulty* (D) is defined as the inverse of program level L:

$$D = \frac{1}{L}. \quad (6)$$

- The program *level estimator* ( $\hat{L}$ ) of L is defined by Halstead as:

$$\hat{L} = \frac{2}{n_1} * \frac{n_2}{N_2}. \quad (7)$$

and interpreted by [Menzies and others, 2002] and by [Fenton and Pfleeger, 1997] as:

$$\hat{L} = \frac{1}{D} = \frac{2}{n_1} * \frac{n_2}{N_2}. \quad (7.1)$$

## Halstead's Metrics (cont.)

- The *intelligent content* (**I**) of a program P is a measure of the information content of program P, and is defined as:

$$I = \hat{L} * V. \quad (8)$$

- Programming *effort* (**E**) is a measure of the mental activity required to reduce a preconceived algorithm to a program P. E is defined as the total number of elementary mental discriminations required to generate a program:

$$E = \frac{V}{L} = \frac{n_1 N_2 N \log_2 n}{2 n_2}. \quad (9)$$

In the effort definition, the unit of measurement of E is claimed by Halstead to be an elementary mental discrimination.

- The required programming time (**T**) for a program P of effort E is defined as:

$$T = \frac{E}{S} = \frac{n_1 N_2 N \log_2 n}{2 n_2 S}. \quad (10)$$

where S is the Stroud number, defined as the number of elementary discriminations performed by the human brain per second. The S value for software scientists is set to 18. The unit of measurement of T is the second.

## Halstead's Metrics (cont.)

---

- All the above ten equations are based on the results of  $n_1$ ,  $n_2$ ,  $N_1$ ,  $N_2$ ,  $n_1^*$  and  $n_2^*$ , which themselves are based on a counting strategy to classify the program tokens as operators or operands.
- Unfortunately, there is a problem in distinguishing between operators and operands.
- This problem occurs because Halstead has provided an example [Halstead's book, pp. 6-8] with specific illustrations of operators and operands, but without generic definitions applicable to any program context.
- Therefore, it is important that the counting strategy be clearly defined and consistent, since all Halstead's software science depends on counts of operators and operands.
- However, there is no general agreement among researchers on the most meaningful way to classify and count these tokens.
- Of course, it is to be expected that different counting strategies will produce different values of  $n_1$ ,  $n_2$ ,  $N_1$  and  $N_2$ , and, consequently, different values for the above ten equations.

# Analysis of their Design and Definitions

---

- **Defining the Context:**

- The objective of Halstead's metrics is to measure the following characteristics of a program: *length, vocabulary, volume, level, difficulty* and *intelligence content*.
- In addition, they are used to measuring what is referred to as “other characteristics” of the developer: programming effort and required programming time.
- The last two attributes, which refer to a developer's attributes (programming effort and required programming time), seem to be identical, since ‘effort to write a program’ is similar to ‘required programming time’.

# Analysis of their Design and Definitions (cont.)

---

- **Designing the measurement:**

- **The empirical and numerical worlds and their mapping:**

- The entities that can be used to apply Halstead's metrics are the source code itself or the algorithm of that source code.
- Applying Halstead's metrics to these two entities will produce different values for the same base measures.
- Halstead's metrics are based on two attributes: the number of operators and the number of operands.
- The two attributes can be easily mapped to a mathematical structure by counting the number of operators and operands in the program source code or the equivalent algorithm.

# Analysis of their Design and Definitions (cont.)

---

- The measurement method:
  - To obtain a value for each of Halstead's metrics, ten equations have to be computed.
  - All of these equations (equations 1 to 10) correspond to a 'derived measure', as defined by the international vocabulary of basic and general terms in metrology (VIM) and the ISO 15939.
  - Equation (3) is of a ratio scale type, while equation (5) is of an ordinal scale type, as noted by [Fenton and Pfleeger, 1996].
  - By contrast, Zuse [1998] maintains that equation (1) is of the ratio scale type and equations (2), (3), (6) and (9) are of an ordinal scale type. Moreover, it can be observed that equation (4) is also of the ratio scale type.
  - It is not clear to which scale type equations (7), (8) and (10) belong.

# Analysis of their Design and Definitions (cont.)

---

- Equation (1), the program length (N) is calculated by the addition of the total number of occurrences of operators and the total number of occurrences of operands. However, since their units are different, operators and operands cannot be directly added together unless the concept common to them (and its related unit) is taken into consideration in the addition of these numbers, that is, 'occurrences of tokens': then, the right-hand side of equation (1) gives 'occurrences of tokens' as a measurement unit on the ratio scale:

$$N \begin{array}{l} \text{occurrences} \\ \text{of tokens} \end{array} = N_1 \begin{array}{l} \text{occurrences} \\ \text{of operators} \end{array} + N_2 \begin{array}{l} \text{occurrences} \\ \text{of operands} \end{array} .$$



## Analysis of their Design and Definitions (cont.)

---

- From equation (2), the program vocabulary ( $n$ ) can be constructed by adding the number of distinct operators and the number of distinct operands:

$$n \text{ distinct tokens} = n_1 \text{ distinct operators} + n_2 \text{ distinct operands} .$$

The measurement unit here is 'distinct tokens'. This measurement unit must then also be assigned to the left-hand side of this equation, labeled 'vocabulary', and associating it to the related concepts.

It can be noted that, while the concept of 'length' is associated with a number, the concept of 'vocabulary' is not. Indeed, the program vocabulary ( $n$ ) reflects a different view of program size [Fenton, 1994], and it is a measure of 'the repertoire of elements that a programmer must deal with to implement the program' [Christensen, 1981]. Most probably, an expression such as 'size of a vocabulary' would have been more appropriate.

## Analysis of their Design and Definitions (cont.)

---

- From equation (3), program volume (V) has been interpreted with two different units of measurement; 'the number of bits required to code the program' [Hamer and Frewin, 1982] and 'the number of mental comparisons needed to write the program' [Menzies *and others*, 2002] on the left-hand side of the equation:

$$V \begin{matrix} \text{bits} \\ \text{or} \\ \text{mental} \\ \text{comparisons} \end{matrix} = N \begin{matrix} \text{occurrences} \\ \text{of tokens} \end{matrix} * \log_2 n \begin{matrix} \text{distinct} \\ \text{tokens} \end{matrix} .$$

Thus, there is no relationship between the measurement unit on the left-hand side and those on the right-hand side of this equation. Furthermore, on the right-hand side, the true meaning of the multiplication of 'occurrences of tokens' and 'distinct tokens' is not clear. Such a multiplication would normally produce a number without a measurement unit.

# Analysis of their Design and Definitions (cont.)

---

- Equation (4) gives the definition of the program potential volume ( $V^*$ ), which is a prediction of the program volume:

$$V^* = (2^{\text{potential operators}} + n_2^* \text{ potential operands}) \log_2(2^{\text{potential operators}} + n_2^* \text{ potential operands}).$$

In this equation, the value '2' was assigned to  $n_1^*$ , The measurement unit of the left-hand side is the same as in the previous equation (equation (3)), while there is no recognizable measurement unit for the right-hand side. As in equation (3), such a multiplication would also normally produce a number without a measurement unit.

# Analysis of their Design and Definitions (cont.)

---

- The program level (L) can be calculated using equation (5), in which there is no measurement unit for the left hand-side, either from Halstead himself or from other researchers. In the sense that this is the correct structure for a ratio with the same unit in both numerator and denominator; the end result is therefore a percentage:

$$L = \frac{V^* \text{ bits}}{V \text{ bits}} = \frac{V^* \text{ mental comparisons}}{V \text{ mental comparisons}}.$$

## Analysis of their Design and Definitions (cont.)

---

- For equation (6), the difficulty (D) is a measure of 'ease of reading' and can be seen as a measure of 'ease of writing' as well [Christensen, 1981]. The right-hand side is also a percentage. What the right-hand side of equation (6) means is a riddle, as its associated label on the left-hand side.
- In Equation (7), for the program level estimator ( $\hat{L}$ ), there is no measurement unit for the left-hand side, while the right-hand side consists of a combination of four distinct measurement units. The exact meaning is again a riddle:

$$\hat{L} = \frac{\begin{array}{c} \text{potential} \\ \text{operators} \\ n_1 \end{array}}{\begin{array}{c} \text{distinct} \\ \text{operators} \\ n_1 \end{array}} * \frac{\begin{array}{c} \text{distinct} \\ \text{operands} \\ n_2 \end{array}}{\begin{array}{c} \text{occurrences} \\ \text{of operands} \\ N_2 \end{array}} .$$

# Analysis of their Design and Definitions (cont.)

---

- In equation (8), referred to as the intelligent content of the program (I), there is no measurement unit on the left-hand side. For the right-hand side of this equation, the measurement unit of  $\hat{L}$  – which is not known since it is a combination of units – is multiplied by the measurement unit of V:

$$I = \hat{L} * V^{\text{bits}} = \hat{L} * V^{\text{mental comparisons}}$$

As for equations (6) and (7), the exact meaning of the left-hand side of equation (8) is a riddle if we attempt to interpret this number with measurement units.

# Analysis of their Design and Definitions (cont.)

- Equation (9) is used by Halstead to compute the effort (E) required to generate a program:

$$E = \frac{\text{elementary mental discriminations} \times \text{distinct operators} \times \text{occurrences of operands} \times \text{occurrences of tokens} \times \text{distinct tokens}}{2^{\text{potential operators}} \times \text{distinct operands}}$$

The measurement unit of the left-hand side of this equation, referred to as 'effort', would be expected to be something such as 'hours' or 'days'. Halstead, however, referred to 'the number of elementary mental discriminations' as the unit of measurement for the left-hand side. Next, in the sense that the 'distinct operators', the 'distinct operands' and the 'occurrences of operands' are, in a generic sense, 'tokens', then it can be concluded that the measurement unit of the right hand-side of this equation is a combination of measurement units. Therefore, there is no relationship between the units of measurement of the left-hand and the right-hand sides.

# Analysis of their Design and Definitions (cont.)

- Finally, equation (10) is used to compute the required programming time (T) for the program:

$$T \text{ seconds} = \frac{n_1 \cdot N_2 \cdot N \cdot \log_2 n}{2^{\text{potential operators}} \cdot 18^{\text{psychological moments per second}} \cdot n_2^{\text{distinct operands}}}$$

Again, the measurement unit of the left-hand side, that is, seconds, does not in any way imply the measurement unit of the right-hand side, that is, a combination of many different measurement units. In view of the fact that, Halstead refers to the ‘moments’ in this equations as “the time required by the human brain to perform the most elementary discrimination” [Halstead’s book, 1977, p. 48].



# Discussion and Conclusions

---

we have investigated a well-known set of measures – Halstead’s metrics – by focusing on their design and, in particular, on their measurement units. The following comments can be made about Halstead’s metrics:

- Based on ISO 15939 and the international vocabulary of basic and general terms in metrology (VIM), Halstead’s metrics can be classified as six based measures ( $n_1$ ,  $n_2$ ,  $N_1$ ,  $N_2$ ,  $n_1^*$  and  $n_2^*$ ) and ten derived measures (equations (1) to (10)).
- Halstead has not explicitly provided a clear and complete counting strategy to distinguish between the operators and the operands in a given program or algorithm. This has led researchers to come up with different counting strategies and, correspondingly, with different measurement results for the same measures and for the same program or algorithm.
- There are problems with the units of measurement for both the left-hand and the right-hand sides of most of Halstead’s equations.

## Discussion and Conclusions (cont.)

---

- The implementation of the measurement functions of Halstead's metrics has been interpreted in different ways than the goals specified by Halstead in their designs. For example, the program length (N) has been interpreted as a measure of program complexity, which is a different characteristic of a program [Fenton, 1994].
- Equations (6) and (7.1), using basic mathematical concepts, lead to  $\hat{L}$  being identical to  $L$ ; this point can be clarified as follows:

$$\hat{L} = \frac{1}{D} \quad (6), \quad D = \frac{1}{L} \quad (7.1),$$

$$\hat{L} = \frac{1}{\frac{1}{L}},$$

$$\hat{L} = L.$$

Therefore, using Fenton's description of  $\hat{L}$  [Fenton and Pfleeger book, 1997, p. 251], the program level estimator is identical to the program level.

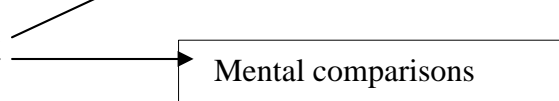
## Discussion and Conclusions (cont.)

- Using the previous observation (that is,  $L = \hat{L}$ ), and from equations (5) and (8), it can be concluded that  $I = V^*$ . The clarification of this point is as follows:

$$I = \hat{L} * V \quad (8), \quad L = \frac{V^*}{V} \quad (5), \quad L = \hat{L} \quad (11),$$

$$I = L * V,$$

$$I = \frac{V^*}{V} \times V,$$

$$I = V^* = \text{size unit}$$


The diagram shows the text 'size unit' from the equation above. Two arrows originate from this text: one points to a box labeled 'Bits' and the other points to a box labeled 'Mental comparisons'.

Therefore, how we can use the same value to measure both ‘intelligent content’ ( $I$ ) and ‘program potential volume’ ( $V^*$ ), two different attributes of a program or algorithm? Also, how do we give different units of measurement to the same value?

## Discussion and Conclusions (cont.)

---

- A number of addition issues can be raised such as the following: Equations (9) and (10), which give the programming effort (E) and the required programming time (T) in seconds, do not take into account technology evolution and characteristics: for instance, new programming languages (i.e. the 4th generation programming languages) need less time for programming since most of the programming effort is expended by means of drag-and-drop processes, as in Visual Basic.
- In summary, the Halstead metrics, as designed almost thirty years ago, do not meet a key design criterion of measures in engineering and the physical sciences.

# Questions?

---



**Merci**



**Thank you**