NIST Technical Note 1990

‘Software Science’ revisited: rationalizing Halstead’s system using dimensionless units

David Flater

This publication is available free of charge from:
https://doi.org/10.6028/NIST.TN.1990

National Institute of
Standards and Technology
U.S. Department of Commerce

NIST Technical Note 1990

‘Software Science’ revisited: rationalizing Halstead’s system using dimensionless units

David Flater
Software and Systems Division
Information Technology Laboratory

This publication is available free of charge from:
https://doi.org/10.6028/NIST.TN.1990

May 2018

U.S. Department of Commerce
Wilbur L. Ross, Jr., Secretary

National Institute of Standards and Technology
Walter Copan, NIST Director and Undersecretary of Commerce for Standards and Technology

National Institute of Standards and Technology Technical Note 1990
Natl. Inst. Stand. Technol. Tech. Note 1990, 12 pages (May 2018)
CODEN: NTNOEF

This publication is available free of charge from:
https://doi.org/10.6028/NIST.TN.1990

'Software Science' revisited: rationalizing Halstead's system using dimensionless units

David Flater

2018-05-08

Abstract

The set of software metrics introduced by Maurice H. Halstead in the 1970s has seen much scrutiny and not infrequent criticism. This article takes a fresh look at these metrics using quantity calculus (the algebra of units) and a new approach to dimensionless units. In this way, it is possible to assign units to the major Halstead metrics in a manner that is logically consistent. However, Halstead's repurposing of counts of software attributes as counts of unobservable mental events leads to a less plausible, more confusing set of metrics for coding effort than for software attributes.

1 Introduction

Elements of Software Science by Maurice H. Halstead [1] is one of the most notable works in the history of software measurement. Published in 1977, it positioned Halstead among the pioneers of the discipline that soon blossomed.

However, there has been strong criticism of both Halstead's theoretical work and of the experimental studies that ostensibly validated it. Sadly, Halstead passed away in January 1979 without responding to these questions and controversies.

The domain of physical science that is most applicable to Halstead's work is metrology, the science of measurement. The International Vocabulary of Metrology (VIM) [2] formally defines metrology terms including quantity, quantity dimension, and measurement unit. A shortened, informal summary of the terms used in this paper is provided in Table 1.

Table 1: Short introduction to relevant terms.

Term	Source	Summary
quantity	[2]	Property of a phenomenon, body, or substance, that can be expressed as the product of a number and a unit of measurement.
unit	[2]	Real scalar quantity with which any other quantity of the same kind can be compared.
dimension	[2]	Expression of the dependence of a quantity on the base quantities of a system of quantities; for example, given the meter as the base quantity of length (L) and the second as the base quantity of time (T), a velocity would have dimension LT^{-1} and be expressible in the unit m/s.
dimensionless unit	[3]	Unit that is regarded as a specialization of the unit 1, having no dependence on any of the seven base quantities of the International System of Units (SI) [4].
counting unit	[3]	Dimensionless unit with which a count of a particular kind of entity or event can be compared.

Recently, this computer scientist revisited Halstead's metrics with the goal of identifying the dimensions or units of each defined quantity using the interpretation of dimensionless units appearing in Ref. [3]. The exercise proved to be unexpectedly fruitful. Once counting units have been assigned to the input quantities, the units of the derived quantities are unambiguously determined. The result could be described as a

rationalization of Halstead's system. However, this process revealed that Chapter 8 of Halstead's book implicitly introduces a parallel set of metrics that are numerically equal to the originals but expressed in less plausible units.

The validity of the metrics is neither demonstrated nor refuted by this analysis. It merely sheds new light on the original definitions, which may then reflect upon the findings of experimental studies.

The remainder of this article proceeds as follows. Section 2 reviews related work and background. Section 3 provides the rationalized definitions and discussion of the metrics themselves. Section 4 identifies problems and concerns that the rationalization did not fix, followed by conclusions in Section 5.

2 Related work

Elements of Software Science has accumulated over 900 citations spanning four decades [5]. Most references to it are in the course of surveying or applying other software metrics. Like McCabe [6], it is part of the canon that must be mentioned.

Halstead's last writings on the subject [7, 8] surveyed contemporaneous efforts at empirical validation. However, later reviewers Hamer and Frewin found that those experiments were poorly designed and inadequate to support the conclusions reached [9]. They also found that the majority of the theoretical assertions in Software Science "represent neither natural laws nor useful engineering approximations."

Shen, Conte, and Dunsmore, in an article with a similar "Software Science Revisited: ..." title, summarized and added to the criticism of the underlying theory that had emerged by the early 1980s [10].

Last but not least, Al-Qutaish and Abran (hereafter abbreviated "A2") performed an analysis of Halstead's metrics with particular attention to units and concluded that they have design flaws [11].[1] The findings most relevant to this article are the following:

- The units of length (N) and vocabulary (η) are "occurrences of tokens" and "distinct tokens" respectively, where "token" is the superordinate concept of both Halstead's "operator" and "operand."
- For volume (V), potential volume (V^*), effort (E), and programming time (T), there is "no relationship" between the units of the input quantities and the units that Halstead claimed for the metrics. They are combinations of disparate quantities that yield results whose units are unclear.
- Program level (L) is a ratio of two quantities of the same kind. However, the estimated program level ($\hat{L}$) is another combination of disparate quantities for which "the exact meaning is again a riddle."
- The scale types (referring to Stevens' taxonomy [13]) of most of Halstead's metrics are unclear.

3 Rationalization

3.1 Input quantities

The input quantities for metrics to be defined in the following subsections are listed in Table 2. They consist of five variables and a constant.

The quantities N_1, N_2, η_1, and η_2 are conceptually straightforward counts of entities in source code, distinguished only by the type of entity counted and by whether duplicates are included or excluded from the count. While determining exactly how to enumerate these entities in a wide variety of programming languages is not as obvious as Halstead thought it would be [14, 15], the theory can proceed on the assumption that such counts are obtainable.

[1] An abbreviated version of the 2005 conference paper by Al-Qutaish and Abran was included as Chapter 7 of Abran's 2010 book, *Software Metrics and Software Metrology* [12].

Table 2: Input quantities for Halstead's metrics.

Name	Symbol	Definition	Unit
Total operators	N_1	"Total usage of all of the operators"	operator
Unique operator count	η_1	"Number of unique or distinct operators"	operator
Total operands	N_2	"Total usage of all of the operands"	operand
Unique operand count	η_2	"Number of unique or distinct operands"	operand
Potential operand count	η_2^*	"Number of *conceptually unique* arguments and results (or input and output parameters) required by a given algorithm"	operand
Stroud number	S	18	mop/s

In A2's analysis, the units of N_1 and η_1 are deemed to be "occurrences of operators" and "distinct operators" respectively, and likewise for the operand counts. We instead find that the counting units are merely operators and operands, and the domain of the count belongs in the definition of the quantity.

The "potential operand count" η_2^* is considerably more difficult to define than the other counts. Halstead used the adjective "potential" in the sense of best possible, so the concept behind η_2^* was to determine the lower bound on the number of inputs and outputs that would be required by an ideal implementation of an operation, one that has no unnecessary data flows. Halstead wrote, "η_2^*, for small algorithms at least, should represent the number of different input/output parameters." However nuanced the scope of the counting might be, the entities counted would seem to be operands in Halstead's model.

Finally, the constant S, although named after John M. Stroud, was assigned a value by Halstead himself. Stroud stated an opinion that "There are approximately ten moments of psychological time for every second of physical time, though there may be more; as many as twenty or less, as few as five." Halstead identified this concept with the average number of "elementary discriminations" that a coder would make per second and assigned the value 18 based on the fit of his model to a sample consisting of twelve machine-language programs [8, 16]. For reasons that will come out in Section 3.5, we replace elementary discrimination with a more general term, mental operation (mop).

3.2 Vocabulary size and program length

The simplest of the derived quantities are vocabulary size (η) and program length (N). Each of these is the sum of two different counts. Following the model of Ref. [3], the unit of such a sum is a generalization of the units of its input quantities. Shen et al., A2, and Zuse [10, 11, 17] each used the name "token" for the superordinate concept. For consistency, we will retain that name, but we again remove the "occurrences" and "distinct" qualifiers from the unit expressions, as this information is included in the descriptions of the quantities. The outcome is shown in Table 3.

Table 3: Vocabulary size and program length.

Name	Symbol	Definition	Unit
Vocabulary size	η	$\eta_1 + \eta_2$	token
Program length	N	$N_1 + N_2$	token

3.3 Program volume

The next quantity is program volume (V, Table 4). An explanation of this quantity was given in 1978 by Fitzsimmons and Love [16]: "For each of the N elements of a program, $\log_2 \eta$ bits must be specified to choose one of the operators or operands for that element. Thus V measures the number of bits required to specify a program." This of course ignores the possibility of a reduction due to data compression or an expansion due to discretization of the number of bits, but the intended binary encoding of tokens is clear.

A2's finding that the units of Halstead's metrics are mysterious follows not only from the different treatment of unique versus total counts, but also from a divergent interpretation of the logarithmic functions. They

Table 4: Program volume, original version and reinterpreted.

Name	Symbol	Definition	Unit
Program volume	V	$N \log_2 \eta$	bit
	V'	$N \log_2 \eta$	mop

credited an email discussion with Richard Peterson of Drexel University when explaining that "In general, in engineering applications we do not take the logarithm of a dimensioned number, only of dimensionless quantities." They then attempted to calculate the logarithm of a length quantity using only mathematical principles.

Rationalizing the logarithmic function instead requires the use of an encoding principle that is widely applied in computer science, as it was by the explanation in Ref. [16]. The unit of V can then be derived from the units of its two terms, N and $\log_2 \eta$:

$$\text{token} \times \frac{\text{bit}}{\text{token}} \rightarrow \text{bit}.$$

Instead of program volume, V would be more accurately described as the *length* of a binary encoding of the program. Given this rationalization of V as a straightforward size metric, a correlation between V and source lines of code (as was reported in Ref. [18] and other sources) should surprise no one.

The confusion about V possibly being a count of mental comparisons rather than bits was created in Chapter 8 of Ref. [1]. There, Halstead introduced a model of mental processes in which the number of mental comparisons (mc) needed for a coder to write a program happened to be $N \log_2 \eta$ (N tokens times $\log_2 \eta$ mental comparisons per token, on the theory that each token was selected through a process akin to binary search).[2] Though numerically equal to the value of V expressed in bits, this is a different kind of quantity, and it was incorrect to conflate it with V. To remedy the confusion, we assign a new symbol (V') to the second, parallel quantity. (Subsequent "parallel" definitions inferred from Chapter 8 will follow this convention of adding a prime to the original symbol.) Similar to the treatment of S, we replace mental comparison with mental operation.

3.4 Potential volume

Halstead uses "potential" in the sense of hypothetical ideal or optimal value; e.g., "the most succinct form in which an algorithm could ever be expressed" in any programming language that one might construct.

As an input to the calculation of potential volume (V^*, Table 5), Halstead derives potential vocabulary (η^*) in a form similar to η. The potential operand count (η_2^*), introduced in Table 2, requires a determination of conceptual uniqueness. The potential operator count, on the other hand, was found to be equal to 2 by reasoning that the lower bound for any algorithm is one operator for the name of the function and another to serve as an assignment or grouping symbol [1, p. 20].

Halstead argued that in the bounding case, neither operators nor operands would need repetition; thus the total operators and operands used equal the unique counts: $N_1^* = \eta_1^*$, $N_2^* = \eta_2^*$, and $N^* = \eta^*$. The derivation of V^* then can be completed similar to that of V, multiplying a number of tokens by a number of bits per token to yield a number of bits.

A2 observes in a footnote that Halstead provides no objective evidence that the hypothetical implementation described is indeed minimal. However, this is a separate matter from rationalizing the quantities that were provided.

[2] Coulter found Halstead's model to be unsupported by cognitive psychology [19].

Table 5: Potential volume.

Name	Symbol	Definition	Unit
Potential operator count	η_1^*	2	operator
Potential vocabulary	η^*	$2 + \eta_2^*$	token
Potential length	N^*	η^*	token
Potential volume	V^*	$\eta^* \log_2 \eta^*$	bit

3.5 Program level and approximated program level

The "level" in Halstead's program level (L, Table 6) metric is used in the sense that it has in the phrases high-level language and low-level language. Derived as the ratio of potential volume to program volume, it reflects the phenomenon that programs written in low-level languages are generally longer than equivalent programs written in high-level languages. However, it is a measurement of the program artifact, not the language. With both V and V^* being measured in bits, L is a ratio of two quantities of the same kind.

An approximation to L ($\hat{L}$) is provided for use when V^* cannot be determined. To be precise, it avoids the need for η_2^* as an input. Unfortunately, when dimensionless units are preserved, we find that $\hat{L}$ is a different kind of quantity than the L that it ostensibly approximates:

$$\frac{\text{operator}}{\text{operator}} \cdot \frac{\text{operand}}{\text{operand}} \to \left(\frac{\text{token}}{\text{token}}\right)^2 \rightsquigarrow \frac{\text{bit}}{\text{bit}}? \,.$$

So, $\hat{L}$ would be better described as a surrogate for L than an approximation to it.

Table 6: Program level.

Name	Symbol	Definition	Unit
Program level	L	$\frac{V^*}{V}$	$\frac{\text{bit}}{\text{bit}}$
Approximated program level	$\hat{L}$	$\frac{2}{\eta_1}\frac{\eta_2}{N_2}$	$\frac{\text{operator}}{\text{operator}} \cdot \frac{\text{operand}}{\text{operand}}$

As a side note, based on an equation that appears in Fenton and Pfleeger [20, p. 251] and in Ref. [21], A2 finds that $L = \hat{L}$ and from this deduces a series of unfortunate conclusions. Having found no basis for the asserted equality in Halstead's book, we believe the cited equation to be apocryphal, an error that was introduced in Ref. [20] (or possibly the first edition of the book) and then propagated from Ref. [20] to Ref. [21]. The cited equation was removed from the third edition (Fenton and Bieman), which defers to Ref. [12] for detailed evaluation of Halstead's metrics [22, pp. 345–346].

Now, as with V, a second quantity is conflated with L in [1, Ch. 8], but this one is not so easy to rationalize. Halstead wrote:

> Each mental comparison requires a number of elementary mental discriminations, where this number is a measure of the difficulty of the task. From the results of Chapter 5, it follows that program level L is the reciprocal of program difficulty.

Since the original L is defined as $\frac{V^*}{V}$, we postulate that this new, parallel interpretation of L, which we call L' (Table 7), should be defined as $\frac{V'^*}{V'}$, with V'^* being another parallel quantity introduced in analogous fashion.

If the above quotation is taken at face value, it seems that L', unlike L, is not a ratio of two quantities of the same kind. If increasing difficulty means an increase in the ratio of elementary mental discriminations (emd) to mental comparisons (mc), it follows that its reciprocal, L', must have the unit mc/emd. But L''s denominator, V', was previously defined to have the unit mc, not emd. To reach the intended result, V'^* would need to have the unit mc^2/emd. How this would arise, given the input quantities and its definition, is a mystery.

At this point it might seem that a more plausible rationalization is to define V' as a count of elementary mental discriminations instead of mental comparisons and allow V'^* to take on the role of the smaller count of mental comparisons. However, this leads to a contradiction later on, in the definition of effort (E).

While it sheds no light on the definitions of elementary mental discriminations and mental comparisons, A2 made the reasonable assumption that V'^* ought to have the same unit as V', i.e., mental comparisons. If we run with this assumption, we discard any distinction that Halstead was trying to make between two different mental operations, but we achieve internal consistency of the metrics. This appears to be the least worst option for rationalizing Chapter 8, and this is why we have substituted mental operation (mop) for both elementary mental discrimination (emd) and mental comparison (mc) in all of the affected metrics.

Table 7: Program level reinterpreted (parallel definitions).

Name	Symbol	Definition	Unit
	V'^*	$\eta^* \log_2 \eta^*$	mop
	L'	$\frac{V'^*}{V'}$	$\frac{\text{mop}}{\text{mop}}$

3.6 Effort

Halstead described the effort metric (Table 8) as "the *total number of elementary mental discriminations* E required to generate a given program."

If we retain the original units that were determined by the derivation of each quantity, we find that E, like V, is a number of bits. It is a greater number of bits than V, since it is scaled up by the reciprocal of the bit ratio L. But this is not the result that Halstead intended.

Introducing E' analogous to the previous definitions leads to a quantity expressed in mental operations.

Table 8: Effort.

Name	Symbol	Definition	Unit
Effort	E	$\frac{V}{L}$	bit
	E'	$\frac{V'}{L'}$	mop

3.7 Estimated implementation time

Having determined E, S is then used as the constant of proportionality to convert it to a quantity of time. The result is estimated implementation time (T, sometimes written $\hat{T}$; Table 9).

To obtain the intended result, it is necessary to use the parallel quantity E' rather than E.

Table 9: Estimated implementation time.

Name	Symbol	Definition	Unit
Estimated implementation time	T	$\frac{E'}{S}$	s

4 Residual issues

In the preceding sections, we had some success in rationalizing some of Halstead's metrics by analyzing them from the perspective of quantities and units. However, there are additional problems that cannot be addressed with this approach. In this section we list them.

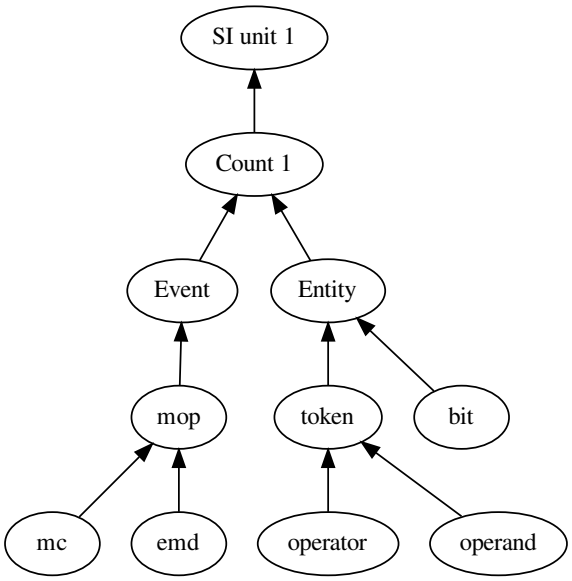

Figure 1: Dimensionless units referred to in this article.

The difference between a mental comparison and an elementary mental discrimination is not clearly explained; neither is it explained why the ratio of the counts of these different mental operations should be numerically equal to L, the ratio of potential volume to program volume.

Use of the mental operation unit (mop) in place of both mental comparison (mc) and elementary mental discrimination (emd) in order to obtain a plausible form for V'^* is a step backward towards treating all dimensionless quantities as if they were mutually comparable. The generalization of operands and operators to tokens was reasoned and deliberate; the generalization of mental comparisons and elementary mental discriminations to mental operations was instead a forced retreat from the more specific expressions that Halstead may have intended.

As A2 observed, all of Halstead's metrics depend on reproducible counting of operators and operands. This reproducibility cannot be achieved through the kind of analysis done in this report, nor indeed from any analysis that is not part of a measurement standard. The consensus and coordination that are essential to established physical metrology are equally necessary to achieve reproducible measurements of these non-physical quantities.

The constant S characterizes the rate of unobservable mental events based on a theoretical model of human thought process. Its value was chosen to obtain the best fit of Halstead's model to a sample of data and was then used forevermore with no associated uncertainty. Both the accuracy of the value of S and the validity of the model to which it is a parameter remain questionable.

Potential volume depends on two questionable inputs. First, the potential operand count (η_2^*) requires a determination of conceptual uniqueness for which no objective procedure has been supplied. Second, the value of η_1^* rests on the questionable assumption that a function must be named. A lambda expression might still require an operator that takes the place of the naming, but the programming language BASIC (for example) allows a whole program to implement a single function without any introduction.

5 Conclusion

This exercise has shown that it is possible to assign units to the major Halstead metrics in a manner that is logically consistent. The type system of dimensionless units that was constructed according to the interpretation of Ref. [3] is shown in Figure 1. The only other unit referenced was the second, as defined in the International System of Units (SI) [4].

As rationalized, all of the metrics produce results that are on a ratio scale. However, Halstead's repurposing of counts of software attributes as counts of unobservable mental events leads to a less plausible, more confusing set of metrics for coding effort than for software attributes.

Although software science has not developed in the way that Halstead envisioned, if software metrics are shown to satisfy the rules of quantity calculus (the algebra of units) instead of only intuition, it is progress.

Acknowledgment

Thanks to Paul Black for helpful suggestions.

References

[1] Maurice H. Halstead. *Elements of Software Science*. Elsevier, 1977.

[2] Joint Committee for Guides in Metrology. *International vocabulary of metrology—Basic and general concepts and associated terms (VIM)*, 3rd edition. JCGM 200:2012, http://www.bipm.org/en/publications/guides/vim.html.

[3] David Flater. Redressing grievances with the treatment of dimensionless quantities in SI. *Measurement*, 109:105–110, October 2017. https://doi.org/10.1016/j.measurement.2017.05.043.

[4] BIPM. *The International System of Units (SI)*, 8th edition, 2006. http://www.bipm.org/en/publications/si-brochure/.

[5] Clarivate Analytics. Web of science, February 2018.

[6] Thomas J. McCabe. A complexity measure. *IEEE Transactions on Software Engineering*, SE-2(4):308–320, December 1976. https://doi.org/10.1109/TSE.1976.233837.

[7] Maurice H. Halstead. Advances in software science. *Advances in Computers*, 18:119–172, 1979. https://doi.org/10.1016/S0065-2458(08)60583-5.

[8] Maurice H. Halstead. Guest editorial on software science. *IEEE Transactions on Software Engineering*, SE-5(2):74–75, March 1979. https://doi.org/10.1109/TSE.1979.234161.

[9] Peter G. Hamer and Gillian D. Frewin. M. H. Halstead's software science—a critical examination. In *Proceedings of the 6th International Conference on Software Engineering (ICSE'82)*, pages 197–206, 1982. https://dl.acm.org/citation.cfm?id=807762.

[10] Vincent Y. Shen, Samuel D. Conte, and H. E. Dunsmore. Software science revisited: A critical analysis of the theory and its empirical support. *IEEE Transactions on Software Engineering*, SE-9(2):155–165, March 1983. https://doi.org/10.1109/TSE.1983.236460.

[11] Rafa E. Al-Qutaish and Alain Abran. An analysis of the designs and definitions of Halstead's metrics. In *15th International Workshop on Software Measurement (IWSM 2005)*, pages 337–352, September 2005. http://profs.etsmtl.ca/aabran/Accueil/AlQutaish-Abran%20IWSM2005.pdf.

[12] Alain Abran. *Software Metrics and Software Metrology*. Wiley, 2010.

[13] Stanley S. Stevens. On the theory of scales of measurement. *Science*, 103(2684):677–680, June 1946. https://doi.org/10.1126/science.103.2684.677.

[14] J.-L. Lassez, D. van der Knijff, J. Shepherd, and C. Lassez. A critical examination of software science. *Journal of Systems and Software*, 2(2):105–112, June 1981. https://doi.org/10.1016/0164-1212(81)90030-3.

[15] A. M. Lister. Software science—the emperor's new clothes? *The Australian Computer Journal*, 14(2), May 1982.

[16] Ann Fitzsimmons and Tom Love. A review and evaluation of software science. *Computing Surveys*, 10(1), March 1978. https://doi.org/10.1145/356715.356717.

[17] Horst Zuse. Resolving the mysteries of the Halstead measures. In *METRIKON Conference*, fall 2005. https://horst-zuse.homepage.t-online.de/z-halstead-final-05-1.pdf.

[18] Meine J. P. van der Meulen and Miguel A. Revilla. Correlations between internal software metrics and software dependability in a large population of small C/C++ programs. In *18th IEEE International Symposium on Software Reliability Engineering (ISSRE)*, 2007. https://doi.org/10.1109/ISSRE.2007.12.

[19] Neal S. Coulter. Software science and cognitive psychology. *IEEE Transactions on Software Engineering*, SE-9(2):166–171, March 1983. https://doi.org/10.1109/TSE.1983.236461.

[20] Norman E. Fenton and Shari Lawrence Pfleeger. *Software Metrics: A Rigorous and Practical Approach*. PWS Publishing Company, 2nd edition, 1997.

[21] Tim Menzies, Justin S. Di Stefano, Mike Chapman, and Ken McGill. Metrics that matter. In *27th Annual NASA Goddard/IEEE Software Engineering Workshop (SEW-27'02)*, pages 51–57, December 2002. https://doi.org/10.1109/SEW.2002.1199449.

[22] Norman Fenton and James Bieman. *Software Metrics: A Rigorous and Practical Approach*. CRC Press, 3rd edition, 2014.