1977

A Quantitative Connection Between Computer Programs and Technical Prose

M. H. Halstead

Report Number:
77-227

A QUANTITATIVE CONNECTION BETWEEN COMPUTER PROGRAMS AND TECHNICAL PROSE

M. H. Halstead
Computer Science Department
Purdue University
West Lafayette, Indiana 47907

CSD-TR 227
April 1977

A QUANTITATIVE CONNECTION BETWEEN COMPUTER PROGRAMS AND TECHNICAL PROSE

M. H. Halstead
Purdue University

ABSTRACT

A basic equation governing programming rates which had been derived and tested on programming tasks requiring less than 100 minutes is shown to agree with recently published Cobol and IBM data of from one to 11,000 man-months. The same theory predicts other relationships which are language independent. These additional relations, when applied to technical English prose, are shown to confirm the theory. This suggests that program specifications or problem statements are now subject to quantitative investigation.

INTRODUCTION

It has been shown [1,3,4] that the number of elementary mental discriminations (E) required to implement a computer program is related to the program's volume (V), the language level (λ), the Stroud Number (S), and the time (T) according to

$$E = ST = V^{1.5} \lambda^{-0.5} \qquad (1)$$

where S = 18 discriminations per second, and λ, while variable, has a mean value near one for common programming languages (1.14 for Fortran, 1.53 for PL/I). The volume (V) is related to the length (N) and the vocabulary (η) through

$$V = N \log_2 \eta = \eta \log_2 (\eta/2) \log_2 \eta \qquad (2)$$

where both η and N are measurable quantities for any available program.

Consequently, if we have both the programming time and the program, we may calculate two independent values of E and compare them. In the absence of the program itself, if we have the number of source statements (P′) we may approximate E by taking average values of three quantities: the language level (λ), the ratio of total source statements (P′) to executable source statements (P) and the number of operands and operators per executable source statement.

THREE DATA SETS

Three recently published sets of data will be used. The first [1] is for small Fortran programs, and includes T in minutes and the number of executable statements P. From this reference itself we can also obtain counts of operands and operators per source statement, or N = 7.5P. The results are shown in Figure 1.

The second set [5] consists of 16 Cobol programs, with T given in man-days, along with the total source statement count (P′). This reference also contains the observation that a program with 1733 statements contained 525 executable ones, or P = 0.303 P′. From reference [2] we can determine that the average executable Cobol statement contains 7.9 operators and operands, or N = 7.9P = 2.39P′. We do not have a mean value of λ for Cobol, and in the absence of the programs themselves we can not calculate it, so we will use the value for PL/I. The results are also shown in Figure 1.

The third data set consists of only the two extreme points cited specifically in [8], but from additional material in that article it appears that they are reasonably consistent with a very large number of intermediate points. Since the language for these points was not specified, λ was taken as the mean between Fortran and PL/I, or 1.34. From Fortran, the rate $N = 7.5P$ was used, as well as the approximation $P = 0.5P'$. The results are shown in Figure 1.

Note that the solid line in Figure 1 extends from five minutes at its lower end to 1000 man-years at its upper end, and that over this range the number of elementary mental discriminations (E) does not differ significantly whether it is calculated from the number of elementary mental discriminations ($E = \sqrt[3]{V^3/\lambda}$), or simply by multiplying the total number of seconds by 18 ($E = ST$).

EXTENSION TO ENGLISH PROSE

While the ability of the effort equation to predict programming times over a wide range of magnitudes may be taken as a triumph of the theory, it also suggests that we must be dealing with rather basic quanta which might be useful in other ways.

For example, since the theory is not restricted to any given language, we might ask if it applies to English. If it can be shown to do so, then a number of potential uses would be apparent. We will approach this task by starting with those relations which are somewhat simpler than equation (1), and testing them against a set of 12 technical abstracts published in a scientific journal [9].

The first requirement is an objective method for identifying operators and operands. Kulm [6] has shown that this requirement is readily met by accepting as operators the list of 363 "Function Words" (Articles, pronouns, prepositions, conjunctions, and auxilliary verbs) published 20 years ago by Miller [7], and adding punctuation, capitalization, and font changes. All other words are then operands.

The second requirement specific to natural language is an objective method of estimating the combined effects of: 1) synonym and antonym usage, 2) variation in tense, and 3) variation in number in the counts of unique operators (η_1) and unique operands (η_2). This can be handled by mechanically counting all unique symbol strings as contributing to either η_1' or η_2', and assuming that each of the three effects contributes one half, so that $\eta' = (1+1/2+1/2+1/2)\eta$ or $\eta = 0.4\eta'$. Total occurrences of operators (N_1) and of operands (N_2) will not be effected by any redundancy.

Using these completely objective techniques, counts of the 12 abstracts yielded the values given in Table 1.

Now the length equation, which has been shown to give a reasonable estimate ($\hat{N}$) of Lengths (N) of computer programs for various languages is

$$\hat{N} = \eta_1 \log_2 \eta_1 + \eta_2 \log_2 \eta_2 \qquad (3)$$

and values of $\hat{N}$ are included.

Table 1. N vs. $\hat{N}$ for technical English.

Abstract	η_1	η_2	N	$\hat{N}$
1	10.4	13.6	84	86
2	16.4	29.6	230	211
3	14.0	24.0	157	163
4	12.4	26.0	179	167
5	15.6	37.6	261	259
6	13.6	33.6	222	222
7	15.6	40.0	306	275
8	17.6	44.8	294	319
9	22.4	60.8	464	461
10	39.4	105.2	874	915
11	14.4	34.4	253	231
12	20.0	43.2	315	321
MEAN			303	302

Clearly, equation (3) predicts the lengths from the vocabularies as well for these technical abstracts as it does for computer programs.

Now for computer programs, if one defines the number of conceptually unique argument and result operands (η_2^*) which a procedure call or subroutine call upon that program would require, then this is related to the language independent potential volume (V*), the volume (V) and the implementation level (L) according to

$$(2 + \eta_2^*)\log_2(2 + \eta_2^*) = V^* = VL \qquad (4)$$

where L may be obtained from $L = 2\eta_2/\eta_1 N_2$. But for English, the very concepts of either η_2^* or V* are undefined. Consequently, while they could be calculated, there is no measureable quantity in these abstracts with which they could be compared. At first this problem seems insurmountable, but an indirect solution is possible. For computer programs, it has been found that η_1, η_2 and η_2^* are related by

$$\eta_2 = A\eta_1 + B$$

where

$$A = \frac{\eta_2^* \log_2(\eta_2^*/2)}{\eta_2^* + 2} \quad ; \qquad B = \eta_2^* - 2A$$

Consequently, by following the six steps given below we can use N alone to obtain estimates of both $\hat{\eta}_1(N)$ and $\hat{\eta}_2(N)$, and compare with measured values.

1) Obtain $\eta(N)$ from $N = \eta \log_2(\eta/2)$.
2) Obtain $V(N)$ from $V = N \log_2 \eta$.
3) Let $\lambda = 1$ in $V^* = \sqrt{\lambda} V$ to obtain $V^*(N)$
4) Obtain $\eta_2^*(N)$ from $V^* = (2 + \eta_2^*)\log_2(2 + \eta_2^*)$.
5) Use $\eta_2^*(N)$ to obtain $A(N)$ and $B(N)$.
6) Obtain $\hat{\eta}_1(N)$ and $\hat{\eta}_2(N)$ from $\eta(N)$, $A(N)$ and $B(N)$.

The results are given in Table 2.

While the agreement found must in some sense be fortuitous, since even in the best experiments, correlations can not be expected to be as high as those found here, it is most difficult to explain it without the assumption that the dual concepts of η_2^* and potential volume apply to technical prose as they do to computer programs.

FIGURE 1. ELEMENTARY MENTAL DISCRIMINATIONS

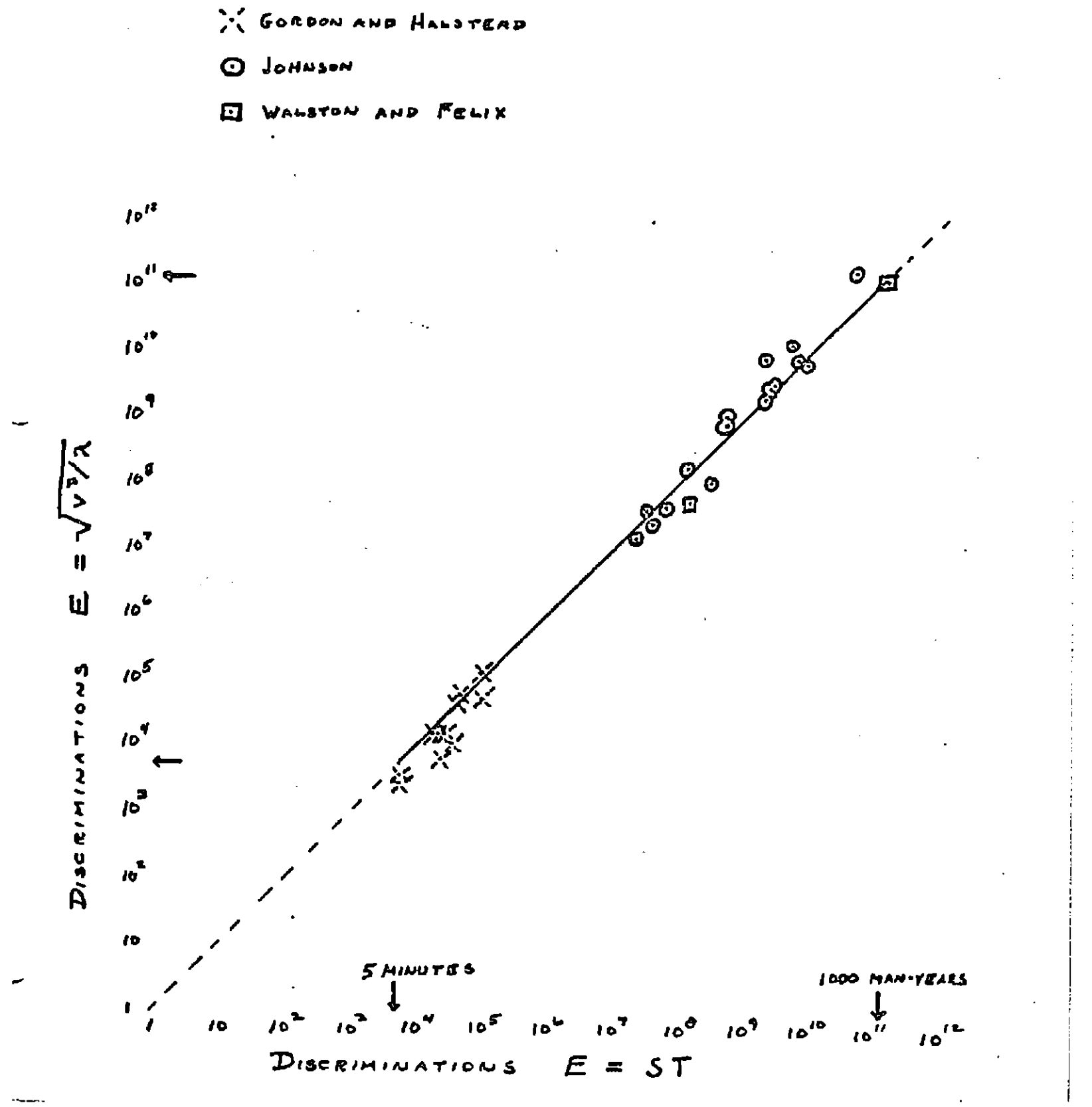

Table 2. Unique Operators and Operands in Technical English, as observed and as Calculated from Length.

Abstract	$\hat{n}_1(N)$	$\hat{n}_2(N)$	n_1	n_2
1	11	13	10	14
2	17	33	16	30
3	14	23	14	24
4	15	26	12	26
5	18	37	16	38
6	16	32	14	34
7	19	43	16	40
8	19	41	18	45
9	24	62	22	61
10	35	107	39	105
11	17	36	14	34
12	19	44	20	43
MEANS	19	41	18	41

REFERENCES

[1] Gordon, R. D. and M. H. Halstead, "An Experiment Comparing Fortran Programming Times with the Software Physics Hypothesis", AFIPS Conference Proceedings, V45, 1976 Nat'l Computer Conf., pages 935-938.

[2] Halstead, M.H., "Software Physics Comparison of a Sample Program in DSL ALPHA and Cobol, IBM Research Report RJ 1460, October 1974.

[3] Halstead, M.H., "Software Physics: Basic Principles", IBM Research Report RJ 1582, May 1975.

[4] Halstead, M.H. Elements of Software Science, Elsevier North-Holland, NY, 1977

[5] Johnson, James R., "A Working Measure of Productivity", Datamation, V23 N2 (February 1977).

[6] Kulm, Gerald, "Language Level Applied to the Information Content of Technical Prose" in Collective Phenomena and the Applications of Physics to other Fields of Science., N.A. Chigier and E. A. Stern, eds. Fayetteville, NY 1975, pages 401-408.

[7] Miller, G.A., E. B. Newman and E. A. Friedman, "Length Frequency Statistics of Written English", Information and Control, V1, 1958 pages 370-389.

[8] Walston, C. E. and C. P. Felix, "A Method of Programming Measurement and Estimation," IBM Systems Journal, V16, N1 (1977) 54-73

[9] EOS, Transactions of the American Geophysical Union, Vol 57, No. 5, May 1976. Twelve abstracts starting with that of K. McCamy on page 402.