1977

Potential Contributions of Software Science to Software Reliability

M. H. Halstead

Report Number:
77-229

POTENTIAL CONTRIBUTIONS OF SOFTWARE SCIENCE TO SOFTWARE RELIABILITY

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

CSD-TR 229
April 1977

229

POTENTIAL CONTRIBUTIONS OF SOFTWARE SCIENCE TO SOFTWARE RELIABILITY

M. H. Halstead
Purdue University

As prominent speakers have been noting for years, whenever a piece of software fails, that failure can not be attributed to a "divide" instruction which has suddenly ceased to divide. Instead, and perhaps more embarrassingly, we must ultimately ascribe the failure to "human error", either in original specification, design, or implementation.

Because all failures must reduce to instances of human error, it follows that to be reliable, software needs only to exhibit freedom from this one source of error. Therefore, to make much progress in the field of software reliability it would appear to be necessary to know, or to know how to calculate, the number of opportunities for error which exist in a given system, and what the effect on this number would be if different methods of implementation had been used.

To start at this point requires in turn that we have knowledge of the quantum of thought, and a process for measuring the number of these quanta required in the implementation of a system. Five years ago such requirements were clearly beyond our reach. Today, however, progress in theoretical and experimental software science [2-5] appears to have placed them within our grasp.

We will illustrate this point by outlining results in the understanding and measurement of effort in program implementation. First, the length (N) of a program is taken as the sum of the usages of

operators (N_1) and operands (N_2) in the program, and its vocabulary (η) is the sum of the number of different operators (η_1) and operands (η_2). These software parameters are related according to

$$N = N_1+N_2 = \eta_1 \log_2 \eta_1 + \eta_2 \log_2 \eta_2 \simeq \eta \log_2 (\eta/2) \qquad (1)$$

It follows from the definitions of length and vocabulary that any program must consist of N non-random selections from a list of η items. Assuming that the selection process approximates a binary search, then the total number of mental comparisons required would equal the program volume (V) which is defined as

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

Now if the difficulty (D) of making one comparison for a given program is defined as the number of elementary mental discriminations required, then the total number of such discriminations (E) must be the product VD. Since V is readily measured, we need only a measure for D. This can be obtained by considering the program level (L).

Note that the highest possible level at which any program could be expressed in any language must be as a procedure call, for any language in which that procedure is already available. Such a call itself has a volume, based upon the number of conceptually unique input-output parameters (η_2^*) and operators (η_1^*) which it requires. Since only a function name and a grouping symbol are required for operators, $\eta_1^* = 2$. Since there will be no repetitions, $N = \eta^*$. Calling this minimum possible volume the potential volume (V^*), it is

$$V^* = (2+\eta_2^*) \log_2 (2+\eta_2^*) \qquad (3)$$

and the level at which a program has been implemented is given by

$$L = V^*/V \tag{4}$$

Since the product LV remains constant as a program is translated from one language to another, it follows that as the volume goes up, the level goes down, in complete accord with our intuitional concept of level. Consequently, this measure of level must also be the inverse of difficulty. The total number of elementary mental discriminations (E) is therefore the simple quotient

$$E = V/L \tag{5}$$

Now we denote the number of elementary mental discriminations made by the human brain by the Stroud number (S), where S has been found to reach "twenty or a little less" [7] discriminations per second. From the implementation time (T) we therefore have a completely independent measure of E, or

$$E = ST \tag{6}$$

If, over a reasonable range, these two measures agree, then it follows that out starting point is secure, and that a software analysis is capable of determining the number of discriminations required to implement a given program.

For small programs, requiring from five minutes to an hour and a half to implement, results have previously been published [3]. Recently, two much larger sets of data [6,8] have been published, both of which contain values of implementation times (T) and total source statements (P'), making it possible to extend the range of comparison of equations (5) and (6). In both cases, values of E = ST may be obtained directly from the published data. For Johnson's data [6],

times are given in man-days. S is therefore 18 discriminations per second times (60 x 60 x 8) seconds per day. Walston and Felix [8] give implementation times in man-months, so S = (18 x 60 x 60 x 2000/12) discriminations per man-month.

In order to evaluate E = V/L, in the absence of the program itself, requires the introduction of another software parameter, the language level (λ), defined as

$$\lambda \equiv LV^* = L^2V$$

where λ has been found to be reasonably conservative with variations in program size, and to have an average value near one for most programming languages. (λ = 1.14 for Fortran, 1.53 for PL/I) [5]. Minor variations in λ will have little effect upon the results, because it will enter only as the square root, and the square root of a number near one is even closer to one. For Johnson's data, which are in COBOL, and Walston and Felix's data, which cover various languages, we will merely use the mean between Fortran and PL/I, or λ = 1.34. From the number of executable source statements (P), it is possible to obtain the length (N). From [3] we have N = 7.5P for Fortran, and from [4] we can obtain N = 7.9P for COBOL. From Johnson [6] we have $P = (525/1733)P'$, so $N = 2.39\ P'$, and for Walston and Felix's data we will use $P = 0.5\ P'$, or $N = 3.75\ P'$. From N, we may use equation (1) to obtain η and with η and N, equation (2) yields V directly.

The calculations are given in the appendix, and shown in Figure 1. When it is noted that the data cover the range from five minutes to 1000 man-years, the linearity observed in Figure 1 can be taken as evidence that for most programs of interest, the number of elementary discriminations used in implementing a program varies directly with the volume (V) and inversely with the level (L).

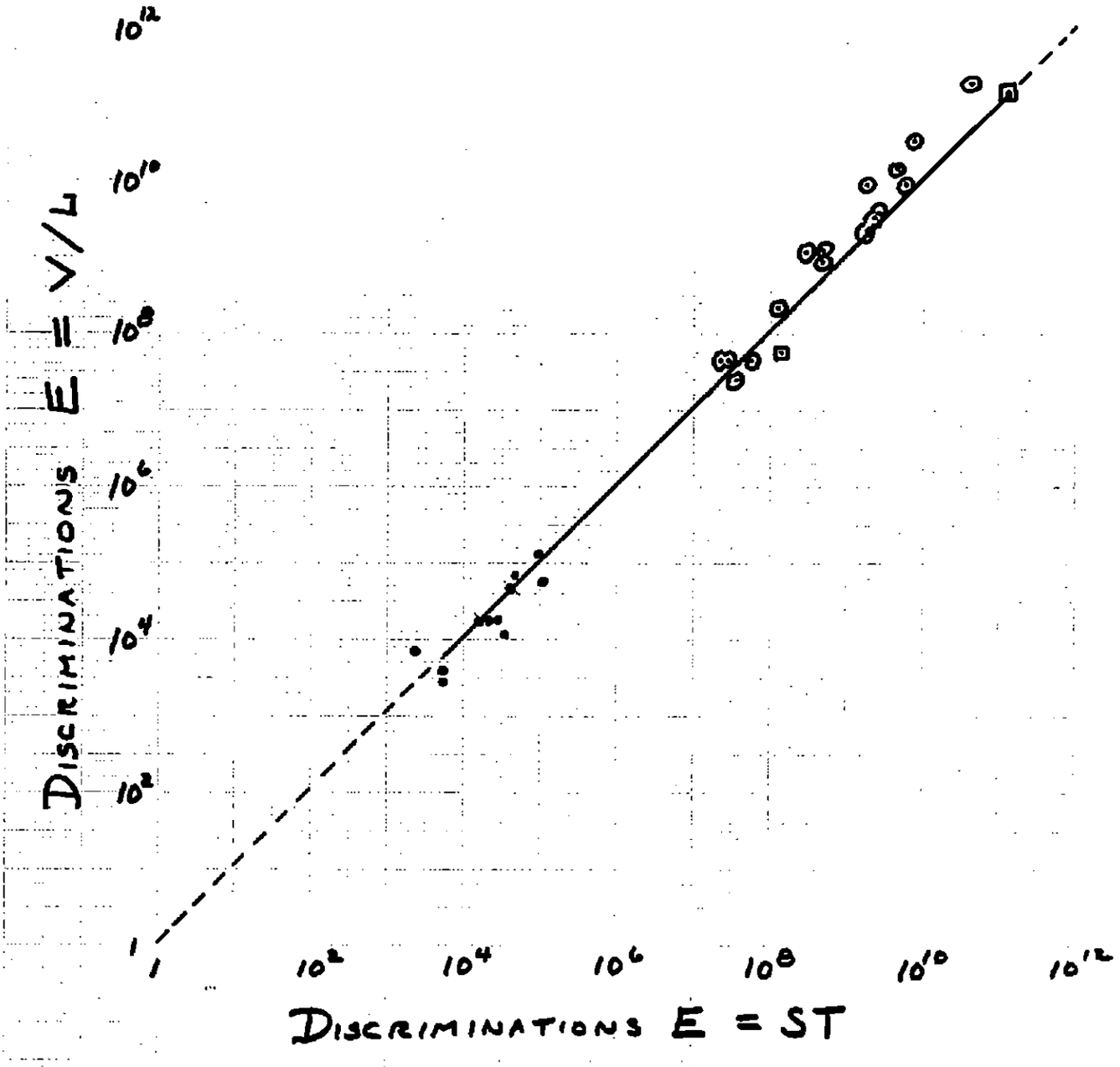
FIGURE 1. ELEMENTARY MENTAL DISCRIMINATIONS

• GORDON-HALSTEAD DATA

⊙ JOHNSON DATA

⊡ WALSTON-FELIX DATA

SOLID LINE EXTEND FROM 5 MINUTES TO 1000 YEARS.



It is then an easy step to the hypothesis that the number of human errors in a system's implementation will depend almost exclusively upon the number of human discriminations performed during that implementation. And in fact, even before the generality demonstrated above was known, this has already been shown [2] to be the case for one large system [1], in which the pearsonian coefficient of correlation between E an the number of reported errors was found to be 0.982.

Potential For Improvement

If we consider the preceding development as providing a base for the understanding of software reliability, rather than merely as a method for measuring or predicting it, then it offers some useful insight.

For example, once complete program specifications are available, the number of conceptually unique input and output parameters (η_2^*) has been set, and it follows that the potential volume (V^*) has also been set. From V^*, the product of volume and level is also set, but not their quotient. In other words, complete program specifications leave us with

$$LV = V^* = \text{Determined}$$

$$V/L = E = \text{Undetermined}$$

By substituting above, we have

$$E = V^2/V^* = V^*/L^2$$

from which it is immediately apparent that any technique which will reduce the program volume will be highly significant in reducing the

mental work, and the "human error" rate. But let us note that while any methodology which reduces volume must also increase the level, it is easier to work first with the level. This results from the fact that level, defined as V^*/V, has also been found to be closely approximated by

$$L = \frac{2}{\eta_1} \frac{\eta_2}{N_2}$$

In this form, it is easy to observe that η_1, which includes one for each different go to, is in the denominator. Consequently, unwise use of go to's, either because they are required by a given language, or because of an idiosyncrasy of style, will decrease level and increase both effort and errors. Similarly, a language which requires constant repetition and reuse of the same operands must contribute to the error rate.

In summary, program implementation error rates can be reduced, and reduced importantly, by the use of higher and higher level languages. If, then, the highest level language available has a compiler which produces object code whose efficiency is inadequate, then the solution is not to accept a lower level language with the increased effort that implies, but to devote the additional effort to improving the compiler.

REFERENCES

[1] Akiyama, F., "An Example of Software System Debugging", Proc. Int'n'l Federation of Information Processing Societies Congress, 1971. North-Holland Publishing, Pages 353-359.

[2] Funami, Yasuo, and M. H. Halstead, "Software Physics Analysis of Akiyama's Debugging Data", in Computer Software Engineering, J. Fox, ed., Polytechnic Press, NY 1976, Pages 133-138.

[3] 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 935-138.

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

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

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

[7] Stroud, John M. "The Fine Structure of Psychological Time", Annals of the New York Academy of Sciences, 1966, Pages 623-631.

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

APPENDIX

Data Set 1. Gordon-Halstead Observations.

(P and T from Table II [3])

P	T (MIN)	N (from P)	η (from N)	V (from N,η)	L (from V,2)	$E=\frac{V}{L}$	E=ST
7	5	53	17	2.17×10^{2}	7.25×10^{-2}	2.99×10^{3}	5.40×10^{3}
8	5	60	19	2.55×10^{2}	6.69×10^{-2}	3.81×10^{3}	5.40×10^{3}
11	21	83	23	3.75×10^{2}	5.51×10^{-2}	6.81×10^{3}	2.27×10^{3}
15	30	113	20	5.49×10^{2}	4.56×10^{-2}	1.20×10^{4}	3.24×10^{4}
18	16	135	33	6.81×10^{2}	4.09×10^{-2}	1.67×10^{4}	1.73×10^{4}
18	19	135	33	6.81×10^{2}	4.09×10^{-2}	1.67×10^{4}	2.05×10^{4}
18	24	135	33	6.81×10^{2}	4.09×10^{-2}	1.67×10^{4}	2.59×10^{4}
32	39	240	51	1.36×10^{3}	2.90×10^{-2}	4.69×10^{4}	4.21×10^{4}
36	92	270	56	1.57×10^{3}	2.69×10^{-2}	5.84×10^{4}	9.94×10^{4}
38	43	285	58	1.67×10^{3}	2.61×10^{-2}	6.40×10^{4}	4.64×10^{4}
59	91	443	83	2.82×10^{3}	2.01×10^{-2}	1.40×10^{5}	9.83×10^{4}

Data Set 2. Johnson Observations.

(P' and T from Table on page 109 of [6]).

P'	T (DAYS)	N (from P')	η (from N)	V (from N,η)	L (fromV,λ)	E=V/L	E=ST
3.1×10^{3}	4.0×10^{1}	7.41×10^{3}	8.49×10^{2}	1.45×10^{5}	3.04×10^{-3}	4.77×10^{7}	2.07×10^{7}
5.6×10^{3}	5.6×10^{1}	1.34×10^{4}	1.42×10^{3}	1.40×10^{5}	3.09×10^{-3}	4.53×10^{7}	2.90×10^{7}
4.0×10^{3}	7.0×10^{1}	9.56×10^{3}	1.06×10^{3}	9.61×10^{4}	3.73×10^{-3}	2.58×10^{7}	3.63×10^{7}
5.9×10^{3}	1.1×10^{2}	1.41×10^{4}	1.48×10^{3}	1.48×10^{5}	3.01×10^{-3}	4.92×10^{7}	5.70×10^{7}
1.4×10^{4}	2.4×10^{2}	3.35×10^{4}	3.15×10^{3}	3.89×10^{5}	1.86×10^{-3}	2.09×10^{8}	1.24×10^{8}
1.0×10^{4}	5.28×10^{2}	2.39×10^{4}	2.34×10^{3}	2.67×10^{5}	2.24×10^{-3}	1.19×10^{9}	2.72×10^{8}
4.1×10^{4}	9.24×10^{2}	9.80×10^{4}	8.17×10^{4}	1.27×10^{6}	1.03×10^{-3}	1.23×10^{9}	4.79×10^{8}
3.4×10^{4}	9.24×10^{2}	8.13×10^{4}	6.92×10^{3}	1.04×10^{6}	1.14×10^{-3}	9.12×10^{8}	4.79×10^{8}
6.0×10^{4}	3.43×10^{3}	1.43×10^{5}	1.15×10^{4}	1.93×10^{6}	8.33×10^{-4}	2.32×10^{9}	1.78×10^{9}
1.4×10^{5}	3.43×10^{3}	3.35×10^{5}	2.47×10^{4}	4.89×10^{6}	5.23×10^{-4}	9.35×10^{9}	1.78×10^{9}
7.5×10^{4}	3.96×10^{3}	1.79×10^{5}	1.40×10^{4}	2.47×10^{6}	7.37×10^{-4}	3.35×10^{9}	2.05×10^{9}
8.0×10^{4}	4.75×10^{3}	1.91×10^{5}	1.49×10^{4}	2.65×10^{6}	7.11×10^{-4}	3.73×10^{9}	2.46×10^{9}
1.9×10^{5}	8.71×10^{3}	4.54×10^{5}	3.25×10^{4}	6.80×10^{6}	4.44×10^{-4}	1.53×10^{10}	4.52×10^{9}
1.4×10^{5}	1.06×10^{4}	3.35×10^{5}	2.47×10^{4}	4.89×10^{6}	5.23×10^{-4}	9.35×10^{9}	5.50×10^{9}
1.3×10^{5}	1.48×10^{4}	3.11×10^{5}	2.30×10^{4}	4.51×10^{6}	1.16×10^{-4}	3.89×10^{10}	7.67×10^{9}
9.1×10^{5}	7.92×10^{4}	2.17×10^{6}	1.35×10^{5}	3.70×10^{7}	1.90×10^{-4}	1.95×10^{11}	4.11×10^{10}

Data Set 3. Walston and Felix Observations.

(P' and T for the two extremes, as cited on page 57 of Reference [8]).

P'	T (MONTHS)	N (from P')	η (from N)	V (from N,η)	L (from V,λ)	E=V/L	E=ST
4.0×10^{3}	1.2×10^{1}	1.50×10^{4}	1.56×10^{3}	1.59×10^{5}	2.90×10^{-3}	5.48×10^{7}	1.30×10^{8}
4.67×10^{5}	1.18×10^{4}	1.75×10^{6}	1.11×10^{5}	2.93×10^{7}	2.14×10^{-4}	1.37×10^{11}	1.27×10^{11}