1977

Further Validation of an Error Hypothesis

Linda M. Ottenstein

Report Number:
77-252
FURTHER VALIDATION OF AN ERROR HYPOTHESIS

Linda M. Ottenstein

Computer Sciences Department
Purdue University
West Lafayette, Indiana 47907

CSD-TR 252
November 1977
FURTHER VALIDATION OF AN ERROR HYPOTHESIS

An earlier paper has presented an hypothesis to predict the number of bugs in a program module [OSH 77]. The estimate for the number of bugs, $§$, is shown to be related to $E$, the number of elementary mental discriminations needed to write the module. The relationship is shown to fit data published by Akiyama [Aki 71] and Bell and Sullivan [BeS 74]. When that paper was written, however, relevant data contained in a careful study done by Shooman and Bolsky was omitted [ShB 75]. In this paper, the hypothesis is tested on that data.

The information presented by Shooman and Bolsky was gathered from the test and integration phase of a moderate sized control type program designed to interphase with many other programs in a large system. It was written in a special purpose language which is essentially an assembly language with powerful macro features added.

Because the data comes from the test and integration phase, the errors reported should be consistent with the concept of delivered bugs. Thus, this data is appropriate for the hypothesis

$$§ = E^{2/3}/3000.$$  

to be tested upon it [OSH 77].

The following subset of hypotheses from software science will be used in the development which follows [Hal 77]:

- **Estimated program length**, $N = \eta \log_2(\eta/2)$  \hspace{1cm} (1)
- **Program volume**, $V = N \log_2 \eta$  \hspace{1cm} (2)
- **Minimum program volume**, $V^* = (\eta_1^*+\eta_2^*)\log_2(\eta_1^*+\eta_2^*)$  \hspace{1cm} (3)
- **Program level**, $L = V^*/V$  \hspace{1cm} (4)
- **Language level**, $\lambda = LV^*$  \hspace{1cm} (5)
- **Number of mental discriminations**, $E = V/L$  \hspace{1cm} (6)
- **Implementation time** $T = E/S$  \hspace{1cm} (7)

where
\( \eta_1 \) = number of unique operators used in a program
\( \eta_2 \) = number of unique operands used in a program
\( \eta_1^* \) = minimum number of unique operators needed to express the algorithm in a potential language (a procedure call)
\( \eta_2^* \) = minimum number of conceptually unique operands needed to express the algorithm in a potential language (a procedure call)

\( S \) = the Stroud number \( \approx 18 \) elementary mental discriminations per second.

Although the numbers of operators and operands are not available, one can estimate \( N \) from the program length, \( P \). It has been shown that

\[
N = (8/3) \times P
\]

for an assembly language program \([\text{Hal} 77]\). Since the language used in the study was not truly an assembly language, the constant \( 8/3 \) might be low. However, it can not be too far off since for Fortran, one would use \( 7.5 \). The substitution of the value \( P = 4000 \) given by Shooman and Bolsky into the equation above results in

\[
N = 10700.
\]

Now we can calculate \( \eta \) from (1), obtaining

\[
\eta = 1180.
\]

Substituting into (2),

(4) giving

\[
V = 109000.
\]

Next, (5) is solved for \( V^* \) which is then substituted into (4) giving

\[
L = \lambda / (L \times V) \text{ or } L = (\lambda / V)^{1/2}.
\]
Finally, using $\lambda = .88$ for assembly language programs [Hal 77], (6) is transformed to:

$$E = \sqrt[3]{\frac{3}{2}} / \lambda^{1/2} = 38,200,000.$$  

Again the value of $\lambda = .88$ might be a little low for the language used in the study, however, the value for Fortran is not much higher. Finally

$$B = \frac{E^{2/3}}{3000} = 38.$$  

This estimate of $B$ is within 20% of the published value of 45. The language used in the study is not truly an assembly language, therefore we can not expect to make a better prediction for $B$. This is partly because twice in our calculations, for lack of better values, we had to use constants based on estimates for assembly language programs.

Shooman and Balsky also mentioned that the average time to find and correct a bug was 4.5 hours. This means that the total debugging time was approximately 203 hours. It is possible to estimate the total implementation time from (7). Then by assuming that 40% of this development time is debugging time, it is possible to calculate an estimate for the debug time (again solely from $P$) to compare with the measured value. The value determined by these calculations is 236 hours, a fairly accurate estimate of the measured value of 203 hours.

CONCLUSION

One more data point does not prove the model. It represents, however, another type of programming application from yet another source for which the model appears to hold. Again, more research in the area is indicated.

---

1Data presented by Barry Boehm indicates that the amount of time needed for check-out and testing is 45% - 50% [Boe 73]. Wolverton's data indicates that this percentage is closer to 36% [Wol 74]. These figures do not seem to dispute the 40-20-40 rule of thumb which states that analysis and design account for 40%, coding and debugging is 20%, and test and integration is 40%.
REFERENCES


[OSH 77] Ottenstein, Linda M., Victor B. Schneider, Maurice H. Halstead, "Predicting the Number of Bugs Expected in a Program Module," CSD-TR 205, Purdue University, January 1977.
