IDEA GROUP PUBLISHING

701 E. Chocolate Avenue, Suite 200, Hershey PA 17033, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com

ITP4745

The "Software Equation" Revisited

Keith B. Olson, Ph.D.
Utah Valley State College, Department of Computer Science
800 West University Parkway, Orem, Utah 84058-5999
Tel: 801-863-6392 Fax: 801-224-2934
Email: olsonke@uvsc.edu

ABSTRACT

n 1992, an empirical "Software Equation" was proposed [2]. The highly non-linear nature of this equation gives rise to some results that are contrary to what one would logically expect. The reasons for this are examined, and an alternative model is proposed.

INTRODUCTION

The Software Equation. Based on data collected from over 4000 software development projects, Putnam and Myers [2] developed an estimation model relating development effort (E, in person-months, for example), time (t, in units to agree with those used on E), and lines of code (L). The equation includes two additional constants that are a function of the particular development environment. The first is B, which is a special skills factor that relates to the process and support structures present in the organization and also to the capabilities required by the particular program. The second is P, which is a productivity parameter, characteristic of the people involved in development. With these definitions, the equation can be stated:

$$E = [L \ B^{1/3}/P]^3 \ t^{-4}.$$

If B and P are combined into a single constant $Q = P/B^{1/3}$, we can write:

$$E = L^3/(Q^3t^4).$$

Pressman [1, page 171] gives the following example that illustrates the nonlinear nature of this model. (Note that Pressman's use of the constant P coincides with the Q used above, not with the P as used by Putnam and Myers.)

"Consider a complex, real-time software project estimated at 33,000 LOC, 12 person-years of effort. If eight people are assigned to the project team, the project can be completed in approximately 1.3 years. If, however, we extend the end-date to 1.75 years, the highly nonlinear nature of the model . . . yields:

$$E = L^3/(P^3t^4) = \sim 3.8 \text{ person-years}$$

This implies that, by extending the end-date six months, we can reduce the number of people from eight to four! [sic]"

The result is actually less logically appealing than Pressman indicates. Since the 3.8 is in units of person-years rather than persons, the actual number of persons required would be 3.8 person-years/1.75 years = 2.17 persons. This model would then say that if we increase the time by about 30%, the number of people will be cut to one-fourth of the number originally required. This is certainly counter-intuitive.

The Root of the Problem. In examining the original derivation of the model by Putnam and Myers, one discovers the basic assumption that productivity is proportional to the effort times the time spent (P = Et). At first glance, this appears to be logical. However, if we recognize that effort is in units of person-time, then the units on the productivity P would be person-$time^2$. This would imply from the outset that if we double the amount of time we spend, we will *quadruple* the amount of code we can produce. If one begins with the assumption that code production is non-linear in time, then it naturally follows that the model will also be non-linear, and will produce the kind of results illustrated above.

A NEW MODEL

The First Approach. A more reasonable model would begin from the assumption that the productivity, in lines of code, would be related to effort by:

$$LOC = kNt$$

where N is the number of people involved in the development, t is the time spent, and k is a proportionality constant. This linear relationship would imply, for example, that if we double the number of people on the team, we will double the output. Real world experience would indicate that this is not true; the increase in the output would be something less than a factor of two. This reduction comes about because of increased communication and coordination overhead among other things. Similarly, if we double the amount of time available, we should slightly more than double the amount of code produced, since some of the overhead in the production is not proportional to the time involved. In general, therefore, the relationship would be more like this:

$$LOC = kF(N,t).$$

A logical simplifying assumption would be that the variables in F can be separated so that

$$LOC = kf(N)T(t).$$

In an effort to determine a complete model, we will first hold the time constant and examine the effect of changing the number of people involved. As indicated above, if we increase the number of people, the rate of change of the LOC differs from a constant by a slowly increasing function of N. We will try this form (we abbreviate LOC to L for this and all equations that follow):

$$dL/dN = P - b \ln N.$$

This model has an obvious flaw. There is a value for N where the derivative becomes negative, which implies that additional increases in N will actually *decrease* productivity. Although there are incidents of exactly this behavior for short time periods, it should not be true for long-term projects. We will see later how this can be managed. Solving the equation leads to this model:

$$L = k[(P + b)N - bN \ln N + C] \ T(t)$$

Assuming that no people would imply zero production, the constant of integration (C) would be zero. Without losing any generality, the constant k can be absorbed into the function f(N) yielding

$$L = [(P + b)N - b \ N \ln N] \ T(t)$$

Assuming that T(t) is held constant, we can evaluate the constants P and b by knowing two productivity figures based on different values of N. Consider a situation (admittedly fictitious) where doubling the number of people increases productivity by 90%. If N_0 is the smaller of the two teams, then P and b are related by

$$P = [12.863 + \ln N_0]b$$

If we know productivity for one team, we can then determine actual values for P and b. If a team of 4 (P_0) can produce 2000 lines of code in our fixed time frame, we would determine that P = 513.858 and b = 36.062. (This sets T(t) = 1, with units undefined). If we compare the results of the prediction equation

$$L = [(P + b)N - b\ N \ln N\} = 549.920\ N - 36.062\ N \ln N$$
$$= [549.920 - 36.062 \ln N]\ N$$

with results found by multiplying the production by 1.9 when N is doubled, we obtain this table:

N	LOC	Equation	Error
4	2000	2000	0
8	3800	3799	.03%
16	7220	7199	.3%
32	13718	13598	.9%
64	26064	25596	1.8%

Obviously there is a point where this model will fail, as mentioned above. However, to be able to extrapolate from 4 and 8 people to 64 with only a 2% error is acceptable. If the constants were determined from N = 32 and N = 64, for example, it would perform well at N = 4 and also at N = 256, certainly a wide range.

Now, let us turn our attention to the time function, T(t). A similar model might be considered here, except that now if we double the time, we should expect to more than double the production. Thus, the differential equation for T would be:

$$dT/dt = R + q \ln t$$

giving rise to the following equation:

$$T(t) = (R - q)t + q\ t \ln t.$$

The constants in this equation would be determined in a manner similar to those for f(N). The combination would give the production model

$$L = [(P + c)N - c\ N \ln N][(R - q)t + q\ t \ln t].$$

This can be rewritten in the form:

$$L = (P + c)(R - q)[N - \{c/(P + c)\}N \ln N][t + \{q/(R - q)\}t \ln t]$$

or

$$L = A[N - B\ N \ln N][t + C\ t \ln t].$$

Having three arbitrary constants in the model (A, B, and C) would require three data points for a team in order to establish a predictive equation for future work. Let's use the following data points to evaluate the constants in the equation:

N	t	L
8	1	20
32	1	72
8	4	88

(This data is generated by assuming that doubling N will multiply L by 1.9, and doubling t will multiply L by 2.1) This generates a set of three non-linear equations in A, B, and C, which can be solved. The resulting model is

$$L = 2.8748[N - 0.0627\ N \ln N][t + 0.0721\ t \ln t]$$

We will apply this model to the data points above, along with a few others. The actual values for the other points were computed using the multipliers mentioned above. The results are in the following table.

N	t	L	Calculated L	Error
8	1	20	19.9998	0.001%
16	1	38	38.0007	0.002%
32	1	72	72.0032	0.004%
32	2	151	151.2065	0.014%
8	4	88	87.9961	0.004%
16	4	167	167.1948	0.012%
32	4	318	316.7993	0.378%
64	8	1269	1251.2165	1.401%

As long as one stays within the points used to determine the constants (interpolation), the results are excellent. If one moves outside those limits (extrapolation), the performance deteriorates, but is still acceptable. This is the approach that would be used to avoid the negative derivative problem mentioned above; the constants should be calibrated with values as close as possible to the points where the prediction is needed.

Let us apply this model to the example quoted from Pressman above. Since we only have one data point available to work with, we will estimate two of the parameters, then determine the third based on the given data. Note at the outset that for eight people to produce twelve person-years of effort will require 1.5 years, not 1.3 as asserted in the example. We will use as a base point the values of N = 8, t = 1.5, and L = 33000. We will approximate B as 0.1 and C as 0.2. Using these values, we determine the equation as:

$$L = 3212\ [\ N - 0.1\ N \ln N][t + 0.2\ t \ln t].$$

Now, if we keep L at 33000 and change t to 1.75, we obtain this equation in N:

$$N - 0.1\ N \ln N - 5.2799 = 0.$$

Solving this by Newton's method, we obtain a value of 6.5 for N, certainly a more reasonable figure. If we use values of B = 0.2 and C = 0.1 instead, the predicted value for N is 6.2. If B = C = 0.2, then the predicted N is 6.1. If we use the values of B and C obtained with the artificial data above, that is B = 0.06 and C = 0.07, then the determined value of N is 6.7, which seems to be a very reasonable figure. It should be noted that 7 people for 1.75 years gives 12.25 person-years of effort, close to the asserted 12 person-years estimated at the outset.

The Second Approach. As indicated above, there is a problem with the model just described. At some point, the logarithm of N becomes large enough that the derivative of the productivity becomes negative. In an effort to eliminate this problem, let us assume an equation of the form:

$$dL/dN = Pe^{-kN}.$$

Solving this equation yields this formula for the productivity:

$$L = (P/k)(1 - e^{-kN}).$$

If we determine the constants P and k by the use of L(4) and L(8) as used previously, we obtain the following equation:

$$L = 20000 (1 - e^{-0.02634N}).$$

Applying this formula to the data previously used, we obtain these results:

N	LOC	Equation	Error
4	2000	2000	0.0%
8	3800	3800	0.0%
16	7220	6878	4.7%
32	13718	11391	17.0%
64	26064	16294	37.5%

A quick glance at the error figures will convince us that this model is not as consistent as the first one presented.

CONCLUSIONS

It is obviously possible to define a predictive model with an equation of the form

$$L = A [N - B N \ln N][t + C t \ln t].$$

In interpolation and extrapolation modes, the model works with acceptable accuracy for the data used. The model has not been initialized with actual industrial data, which is an essential test for validity. A significant problem with the model is that three data points are required in order to determine the constants.

The second approach given above seems less promising, although it does have features that would recommend it.

Future Work. This is obviously a work in progress. One significant step that needs to be taken is to determine the constants using industrial data. That is almost certainly the best test of its validity. There are other models that could and should be examined, particularly those that will avoid the negative derivative problem of the first model.

There is another problem with this model that requires some attention. It is not really possible to solve the equation explicitly for N in terms of L and t. Thus, finding N when L and t are given requires a numerical approach such as Newton's method. The same is true when solving for t given N and L. This is not a serious difficulty, but does detract from the elegance of the theory.

In spite of these problems, the usefulness of a software equation of this type makes it worth the effort to find an approach that eliminates the difficulties with the equations used in the past.

REFERENCES

1. Pressman, R., *Software Engineering, A Practitioners Approach, Fifth Edition,* McGraw Hill, 2001.

2. Putnam, L., and W. Myers, *Measures for Excellence*, Yourdon Press, 1992.