Programming Measurement and Estimation in the Software Engineering Laboratory*

Victor R. Basili

Department of Computer Science, University of Maryland

Karl Freburger

General Electric Information Services

This paper presents an attempt to examine a set of basic relationships among various software development variables, such as size, effort, project duration, staff size, and productivity. These variables are plotted against each other for 15 Software Engineering Laboratory projects that were developed for NASA/Goddard Space Flight Center by Computer Sciences Corporation. Certain relationships are derived in the form of equations, and these equations are compared with a set derived by Walston and Felix for IBM Federal Systems Division project data. Although the equations do not have the same coefficients, they are seen to have similar exponents. In fact, the Software Engineering Laboratory equations tend to be within one standard error of estimate of the IBM equations.

INTRODUCTION

Many models of software development have been proposed in the literature. They all assume some set of relationships among the factors affecting the process. One of the goals of the Software Engineering Laboratory (SEL) [1, 2] has been to try to understand the development process by collecting and using data to evaluate the relationships proposed in the various models.

The Software Engineering Laboratory is a joint effort of NASA/Goddard Space Flight Center, Computer Sciences Corporation, and the University of Maryland. Its general goals have been to analyze the development of software in order to evaluate software development practices, models, and metrics so they may be better applied in understanding, managing, and engineering the process and the product. This paper analyzes one particular set of programming factors and their interrelationships. It completes a previous study [3] based on fewer data. These factors include the development effort, lines of code, number of modules, duration, pages of documentation, team size, and productivity. One of the most interesting relationships is between lines of code and effort. Contrary to intuition, previous researchers have reported that the relationship between these two factors is almost linear [4, 5]. Several studies have been conducted on a subset of these relationships.

Chrysler [6] collected data on 36 programs in one organization, Johnson [7] collected data on 169 programs in one organization, and LaBolle [8] and Nelson [9] analyzed data derived largely from the System Development Corporation. Jeffrey and Lawrence [4] present results obtained from the analysis of 103 programs from three organizations. Walston and Felix [5] collected data on 60 projects in one organization. A full discussion of previous programming productivity research may be found in the article by Chrysler [6].

It is clear that because of biases in the data and data collection process and the lack of control in the various studies, including our own, only the collection of data in many environments by many researchers will permit a wealth of evidence to be assembled sufficient to generate confidence in the relationships derived. In order to do this, however, results must be published using agreed upon, well-defined terms and explicitly stated environmental constraints so that the experimenter can relate what he is testing to previous studies.

To evaluate the relationships between factors, we

*This research was supported in part by National Aeronautics and Space Administration grant NSG-5123 to the University of Maryland.

Address correspondence to V. R. Basili, Department of Computer Science, University of Maryland, College Park, Maryland 20742.

0164–1212/81/010047-11$02.75

tried to compare our findings with those of a previous study using the same definitions and trying, whenever possible, to make clear the differences in the two environments. Some variations of these factors were also studied and these are explicitly defined in this paper. The data here were obtained from a set of projects developed at NASA/Goddard by Computer Sciences Corporation. The findings are compared with the results of a study by Walston and Felix [5] at IBM Federal Systems Division. The major differences in the two environments are summarized below.

The SEL data base currently includes 15 projects, ranging in size from 1.6 to 112 thousand lines of code and in effort from 1.8 to 116 man-months. The Walston-Felix IBM data base includes 60 projects ranging in size from 4 to 467 thousand lines of code and in effort from 12 to 11,758 man-months. The IBM project data involve eight different languages on 66 different computers covering a very wide range of applications, personnel, and experience. The SEL projects are all in FORTRAN on two different computers and involve predominantly the functions for ground support software for satellites. Most projects were developed from a reasonably common programming pool, and most of the designs were well understood and similar to work previously performed, if not by the particular individual at least by the organization. In fact, most of the top-level design is somewhat standard.

Further discussion of the SEL data is given in the next section. The third section discusses the basic relationships derived from SEL using the same techniques as those of Walston and Felix [5]. Our data are then fit to the Waltson–Felix equations where appropriate. We conclude by attempting to validate the hypothesis, suggested by Jeffery and Lawrence [4], of a linear relationship between effort and product size.

THE SAMPLE DATA

Fifteen completed projects were used in this study. All were developed at NASA's Goddard Space Flight Center by NASA personnel and outside contractors. Five of the 15 projects are attitude determination systems, developed on an IBM System 360 in FORTRAN with some minor assembly language code. One project is an attitude determination support utility used to calculate parameters needed by the larger attitude determination systems. It was a one-man project developed on a PDP 11/70 and converted to an IBM System 360, which was the operational machine. The seventh project is an interactive graphics package developed on a PDP 11/70 in FORTRAN and MACRO-11 assembly language. A similar system already existed on an IBM System 360. The other eight data points are separately developed subsystems of a single attitude determination system. The individual data points from design through testing represent their independent development profiles and do not include any subsystem integration effort. They were developed for an IBM System 360 in FORTRAN. Each subsystem was developed predominantly by a single individual. More information on several of the projects, methodologies, and environment are given by Basili et al. [1].

Data were collected from these projects with the forms and techniques described by Basili et al. [1]. The data of interest in this study were the total effort required to produce the finished product, lines of source code in the finished product, number of modules, project duration, documentation size, productivity, and average staff size. A detailed description of these follows:

a. The *total effort E* is defined as the number of man-months of effort used on a project, starting when the requirements and specifications become final through acceptance testing. It includes programming effort and managerial and clerical overhead. One man-month of effort is defined here as $173\frac{1}{3}$ man-hours.
b. The total number of *delivered lines L* of source code is defined as the total number of lines of source code delivered as the final product (expressed in thousands of lines). It does not include stubs or any code thrown away. Source lines are 80-character source records provided as input to a language processor, including data definitions and comment lines.
c. The number of *lines of new code* (NL) is the number of source lines in the final product that are not reused code (expressed in thousands of lines). A block of code is considered to be reused if it was developed for a different project and less than 20% of the code is changed; if more than 20% of the code is changed, the whole block is considered new code.
d. The number of *developed lines* of code (DL) is a derived quantity equal to the number of new lines plus 20% of the reused lines: $\text{DL} = \text{NL} + 20\%(L - \text{NL})$. The 20% overhead is charged to account for such items as system integration and full system test.
e. The total number of *modules M* in a project is the number of modules delivered in the final product. For all of the projects in this study, a module is defined as a separately compilable entity, such as a subroutine, function, or BLOCK DATA unit.
f. The number of *new modules* (NM) is the number of modules in the final product that are not reused modules. A module is considered to be reused if it was developed for another project and has less than 20% of its code changed.

g. *Project duration D* is defined to be the time (in months) from the start of a project (receipt of requirements and specifications) to the end of acceptance testing.
h. *Documentation* (DOC) is measured in pages and is defined as the program design, test plans, user's guide, system description, and module descriptions. The program design is a handwritten document. The module descriptions contain a one-page description of each module in the final product.
i. *Productivity P* is a derived quantity, defined as the ratio L/E of total lines of source code to the total effort required to produce the lines of code. Productivity is expressed in lines of code per man month of effort.
j. The *average staff size S* of a project is defined as the total man months of effort divided by the project duration: $S = E/D$.

Because one of the stated objectives of this research was to compare the results with those of a previous study [5], some of the definitions were chosen to be consistent with the definitions used in that study. The definitions of new and developed source lines of code (NL and DL) were selected to match the definitions used in the programming environment under study. The definition of documentation size was constrained by the data available.

ANALYSIS AND RESULTS

This section presents an analysis of various relationships that may be useful as estimating aids to project personnel. The data used for each variable are as described in the previous section. A summary of the results is presented in Table 1 (see also the end of this paper).

Where Walston and Felix performed an analysis of a similar relationship a comparison of the results is given. Where the results of this study and the Walston–Felix study are considerably different, an attempt is made to determine what factors (if any) may contribute to that difference.

To compute the relationships between the variables, two-variable regression is used. For exponential relationships (such as those presented in the Walston–Felix paper), the data are first linearized by taking logarithms. A two-variable linear regression (least squares fit) is then performed on the transformed data. The linear coefficients become exponential relationships when transformed back into the original domain of the data. For linear relationships, a two-variable linear regression is performed on the data.

The standard error of estimate (SE) provides an estimate of the range above and below the line of estimation within which a certain proportion of the items may be expected to fall if the scatter is normal. Assum-

Table 1. Summary of Results

Estimated variable	SEL equation	SE[a]	r**2[b]	Level of significance	Walston–Felix equation	SE[a]	r**2[b]
Total effort	$E = 1.38*(L**0.93)$	1.41	0.93	0.001	$E = 5.2*(L**0.91)$	2.51	0.64
	$E = 1.58*(NL**0.99)$	1.31	0.96	0.001			
	$E = 1.48*(DL**0.98)$	1.29	0.96	0.001			
	$E = 0.652*(M**1.19)$	1.49	0.90	0.001			
	$E = 0.183*(NM**1.05)$	1.57	0.87	0.001			
	$E = 1.04*L + 2.04$	16.1	0.82	0.001			
	$E = 1.55*NL + 1.19$	11.0	0.91	0.001			
	$E = 1.46*DL + 0.22$	10.7	0.92	0.001			
	$E = 0.27*NM - 2.20$	11.4	0.91	0.001			
Productivity	$P = 698*(RNLTOL** - 0.75)$	1.29	0.50	0.01			
	$P = 727*(RNMTOM** - 0.55)$	1.32	0.38	0.02			
Documentation	$DOC = 30.4*(L**0.90)$	1.41	0.92	0.001	$DOC = 49*(L**1.01)$	2.68	0.62
	$DOC = 38.1*(NL**0.93)$	1.52	0.885	0.001			
	$DOC = 34.7*(DL**0.93)$	1.45	0.91	0.001			
	$DOC = 1.54*(M**1.16)$	1.45	0.91	0.001			
	$DOC = 4.82*(NM**0.99)$	1.67	0.83	0.001			
Project duration	$D = 4.55*(L**0.26)$	1.36	0.55	0.01	$D = 4.1*(L**0.36)$	1.72	0.41
	$D = 1.96*(M**0.33)$	1.37	0.54	0.01			
	$D = 4.62*(NL**0.28)$	1.33	0.61	0.01			
	$D = 4.58*(DL**0.28)$	1.34	0.59	0.01			
	$D = 2.5*(NM**0.30)$	1.38	0.55	0.01			
	$D = 4.39*(E**0.26)$	1.37	0.52	0.01	$D = 2.47*(E**0.35)$	1.52	0.60
Staff size	$S = 0.24*(E**0.73)$	1.38	0.89	0.001	$S = 0.54*(E**0.6)$	1.56	0.79

[a]Standard error of estimate.

[b]Coefficient of determination.

ing a normal distribution of the deviations from the estimation line, we may expect to find about two-thirds of the items (ideally 68.27%) within the band +SE to −SE about the line of estimation, about 95% (ideally 95.45%) within the wider band that includes +2*SE to −2*SE, and practically all (99.73%) within +3*SE to −3*SE. The standard error of estimate is a general or overall measure of the dispersion of all of the Y values around the estimating equation but is often used to indicate the dependability of specific estimates.

The coefficient of correlation expresses the degree of relationship between the two variables. The coefficient of correlation varies from +1 to −1. The sign indicates whether the two variables are directly correlated (positive) or inversely correlated (negative), while the magnitude of the coefficient indicates the degree of association. When there is absolutely no relationship between the variables, $r = 0$. A perfect correlation between the variables is indicated when the magnitude of $r = 1$. The coefficient of determination (r**2) is the amount of variation that has been explained by the line of relationship; 1 − (r**2) is that part of the total variation that has not been explained.

Some of the relationships are illustrated by diagrams (for example, see Figure 1). Each + represents the data from one of the completed projects. The solid line is the estimating equation, or line of regression, computed as described above. The broken lines represent bounds of one standard error of estimate from the estimating equation. The estimating equation, standard error of estimate, and coefficient of determination r^2 are shown in Table 1.

Those relationships also studied by Walston and Felix are illustrated by a diagram comparing their estimating equation with the SEL equation. In Figure 2 each + represents the data from one of the completed SEL projects. The solid line represents the Walston–Felix estimating equation. The two broken lines parallel to the estimating equation represent bounds of one standard error of estimate from the Walston–Felix estimating equation. The other broken line (with finer dash structure) represents the SEL estimating equation.

The derived estimating equations could be used in the following manner. After the project estimates have been computed, those estimates can be checked against the equations that provide an independent estimate based on past experience. Project personnel can then compare these with their own estimates. For example, assume that the size of a delivered software product is estimated by project personnel as 100.000 lines of source code and the effort has been estimated as 200 man-months. However, based on the equation in Figure 1, the estimated total effort for a 100.000-line system should be about 100 man-months. The significant difference between the two estimates does not necessarily imply an error on the part of the project personnel, but it does suggest that the assumptions and estimates leading to the project personnel estimate might be reexamined.

The estimating equations presented here should be considered initial approximations, applicable only to the same environment that the subject projects are from. As data for more projects become available, the estimating equations should be updated and refined.

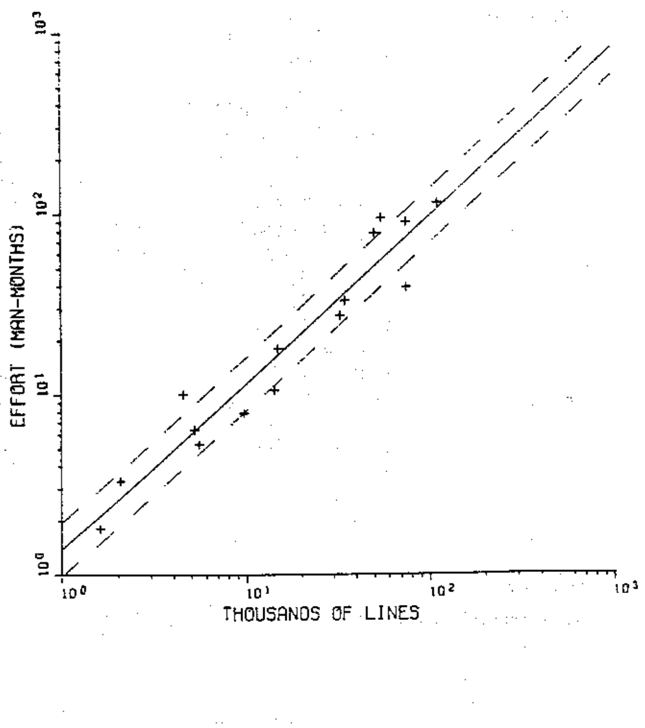
Effort

Effort vs Total Lines. The relationship between delivered source lines of code and total effort is shown in Figure 1. The relationship derived from the data is

$$E = 1.38*(L**0.93). \qquad (1)$$

The standard error of estimate can be used to get bounds on the predicting equation. For example, here the standard error of estimate is 1.41, so the coefficient of the exponential term should be multiplied by 1.41 to get an upper-bound equation and divided by 1.41 to get a lower-bound equation. This gives the equations $E = 1.95*(L**0.93)$ and $E = 0.98*(L**0.93)$ as bounds of one standard error of estimate from the estimating equation (1). The coefficient of determination (r**2) is a significantly high (at the 0.001 level) 0.93, indicating (at least for these projects) that there is a high probability of a relationship between total effort and deliv-

Figure 1. Effort vs lines of code: $E = 1.38*(L**0.93)$.



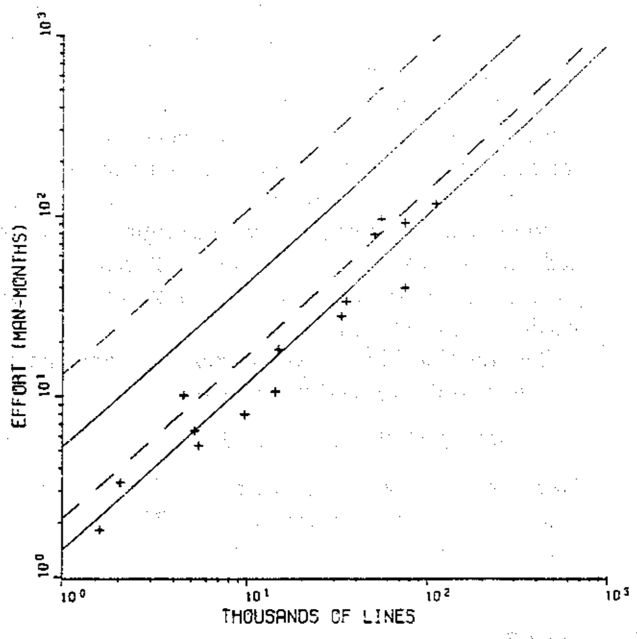


Figure 2. Effort vs lines of code.

ered lines of source code. This relationship is nearly linear.

A linear fit of the data yields

$E = 1.04*L + 2.04.$ (2)

For a linear fit, the standard error of estimate should be added to the constant term. Thus, the standard error of estimate of 16.1 would give the equations $E = 1.05*L - 14.06$ and $E = 1.05*L + 18.14$ as bounds of one standard error of estimate. However, it is not statistically valid to report a standard error of estimate directly from a least squares linear fit since the points are not uniformly distributed around the prediction line. An additive standard error would be unreasonable, since it would be too small for large projects and too large for small projects.

Walston and Felix also found a nearly linear relationship between total effort and product size:

$E = 5.2*(L**0.91)$ (3)

with a standard error of estimate of 2.51. This places the equation derived from the SEL data somewhat below one standard error of estimate of the Walston-Felix equation (see Figure 2). Equation (1) seems to indicate that less effort is required than predicted by (3) to develop the same amount of product. A possible explanation is that the projects studied by Walston and Felix were very diversified; that is, there were many different types of programs [5]. In the SEL environment, however, the programs are almost all of the same general type, and the project personnel have experience developing this type of software, implying there may be less design effort required. In the Walston-Felix study, however, many of the projects were of the large, complex, one-time custom program type where the problems and their solutions are not well understood.

Effort vs New Lines and Developed Lines. Some programming projects reuse code from previous projects in an attempt to reduce the total effort required to produce a system. The relationship between total effort and thousands of new delivered source code,

$E = 1.58*(NL**0.99),$ (4)

is also nearly linear and has a high coefficient of determination.

A linear fit of the data gives

$E = 1.55*NL + 1.19.$ (5)

Substituting developed lines for new lines, the equations become

$E = 1.48*(DL**0.98),$ (6)
$E = 1.46*DL + 0.22.$ (7)

(See Table 1 for standard error and coefficient of correlation values.)

The relationships between total effort and total new and developed lines of source code have high coefficients of determination, indicating that they could be used to predict the total effort if the number of lines of source code (either total or new) could be determined beforehand.

Effort vs Modules. Another measure of program size is the number of modules in the product. Total effort and the number of modules in the delivered product are related as shown in Figure 3:

$E = 0.65*(M**1.19).$ (8)

The relationship is not as linear as that between total effort and delivered lines of source code, but the coefficient of determination indicates there is a high correlation between the total effort and the number of modules in the delivered product. A similar relationship exists between total effort and the number of new modules:

$E = 0.183*(NM**1.05).$ (9)

A new module is defined as a completely new module or one used from a previous project and having more than 20% of the module changed. A linear fit of the data gives

$E = 0.27*NM - 2.20.$ (10)

These equations may be more useful as estimating

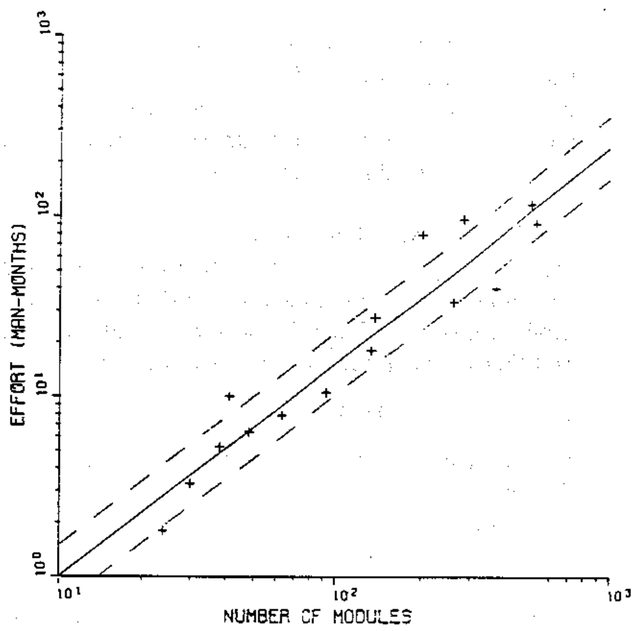


Figure 3. Effort vs number of modules: $E = 0.65^*(M^{**}1.19)$.

aids than (1) and (3) since it is more likely that the number of modules (or a good approximation) is known early in the project life cycle, particularly after the preliminary design phase.

In all the above relationships between effort and size there appears to be a linear relationship independent of the particular size measure. This means that productivity remains relatively constant as the size of the project changes. This may seem surprising, but it does support the IBM Federal Systems result.

Productivity

Productivity is one of the most important factors in all software estimating processes. Here productivity is defined as the ratio of delivered source lines of code to the total effort (in man months) required to produce the product. For this environment, productivity, calculated in terms of delivered lines L, new lines (NL), and developed lines (DL), is in the range of 600–700 lines of code per man-month. It must be remembered, however, that this productivity figure includes the design, code, and testing phases only.

Productivity plotted against the ratio of new lines of source code to total delivered lines of source code produces (see Figure 4)

$$P = 698^*(\text{RNLTOL}^{**} - 0.75). \quad (11)$$

New code is defined as before. The relationship between the two variables suggests that productivity is lowest when there is no reused code. As the percentage of reused code increases, the expected overall productivity increases. This reinforces the intuitive idea that the reuse of a code should be less expensive than creating the code from scratch. The coefficient of determination (0.50) is significant at the 0.01 level.

The Walston–Felix definition of reused code is related more to size change above the original rather than code added, which is significantly different from that used in this paper, so a comparison of the two results would be meaningless.

The relationship between productivity and the ratio of new modules to total modules is

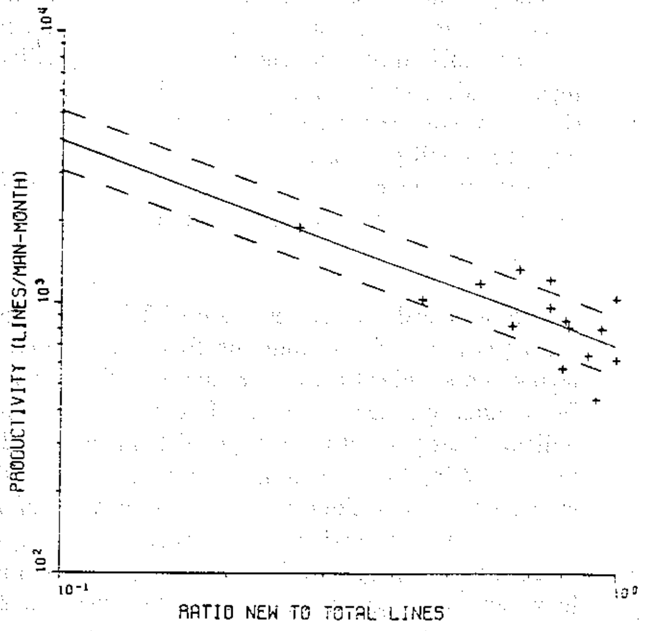
$$P = 727^*(\text{RNMTTOM}^{**} - 0.55). \quad (12)$$

New modules are defined as before. This relationship exhibits the same behavior as (11). The coefficient of determination is significant at the 0.02 level.

Documentation

Documentation is an important part of any software project, and the costs of producing documentation are a factor in the software estimating process. The size of documentation is measured in pages. Here, documentation is defined as the program design (handwritten), test plan, user guide, system description, and module description. The module description contains a one-page description of each module. Figures 5 and 6 show the number of pages of documentation vs thousands of

Figure 4. Productivity vs RNTOL: $P = 698^*(\text{RNTOL}^{**} - 0.75)$.



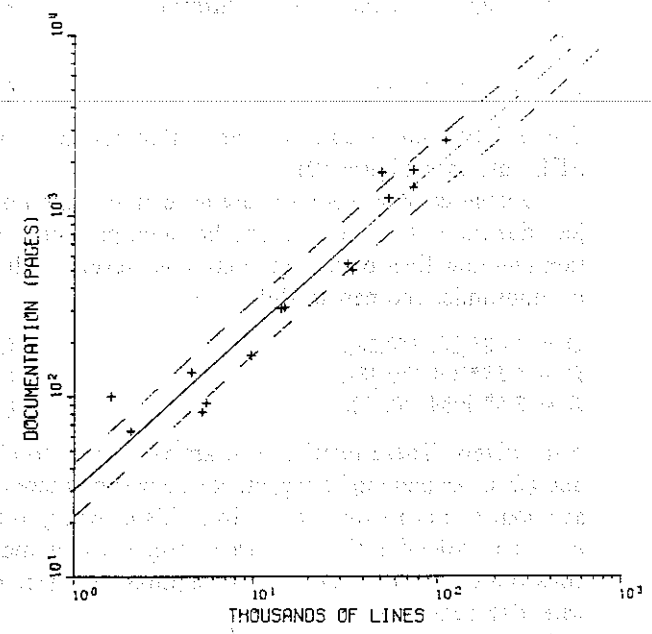


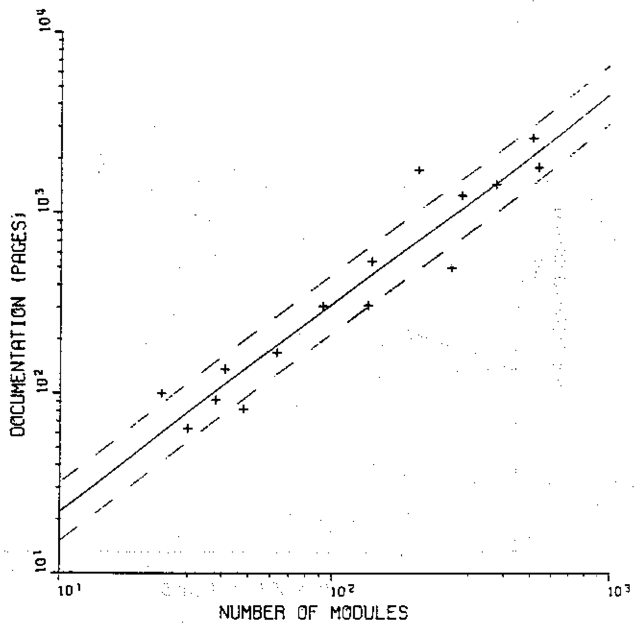
Figure 5. Documentation vs lines of code: DOC = 30.4*(*L***0.90).

delivered lines of source code and number of modules, respectively. The correlation equations are

DOC = 30.4*(*L***0.90), (13)
DOC = 1.54*(*M***1.12). (14)

Both relationships are roughly linear and have coefficients of determination significant at the 0.001 level.

Figure 6. Documentation vs number of modules: DOC = 1.54*(*M***1.12).



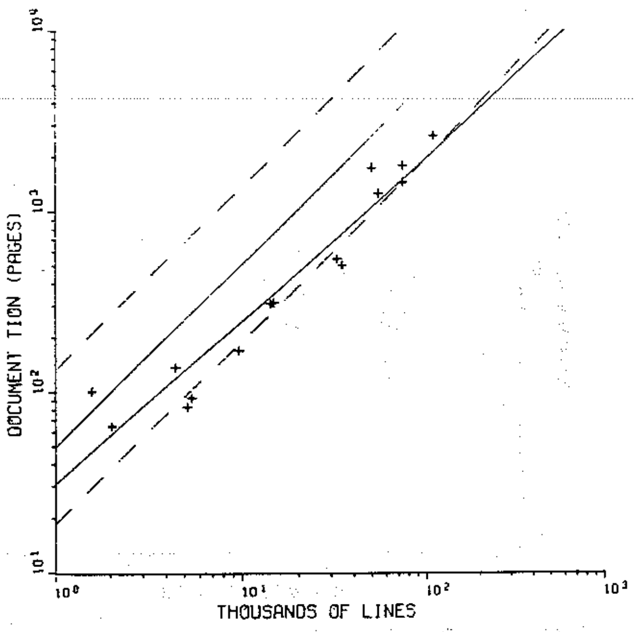


Figure 7. Documentation vs lines of code.

Walston and Felix also found that a nearly linear relationship exists between the number of pages of documentation and the number of thousands of delivered lines of source code:

DOC = 49*(*L***1.01). (15)

Figure 7 shows a comparison of (13) and (15). The SEL equation lies about one standard error of estimate below the Walston–Felix equation. Part of the difference may be explained by the fact that Walston and Felix included in their definition of documentation such items as flowcharts and source program listings, which are not included in the SEL documentation page counts.

Documentation as a function of each remaining size measure, new lines, developed lines, and new modules is

DOC = 38.1*(NL**0.93), (16)
DOC = 34.7*(DL**0.93), (17)
DOC = 4.82*(NM**0.99), (18)

respectively. Again, notice that these relationships are approximately linear. The coefficients of determination are significant at the 0.001 level.

Duration

The problem of determining the duration of a software project is difficult and important. The relationship between project duration (in months) and number of thousands of lines of source code is shown in Figure 8,

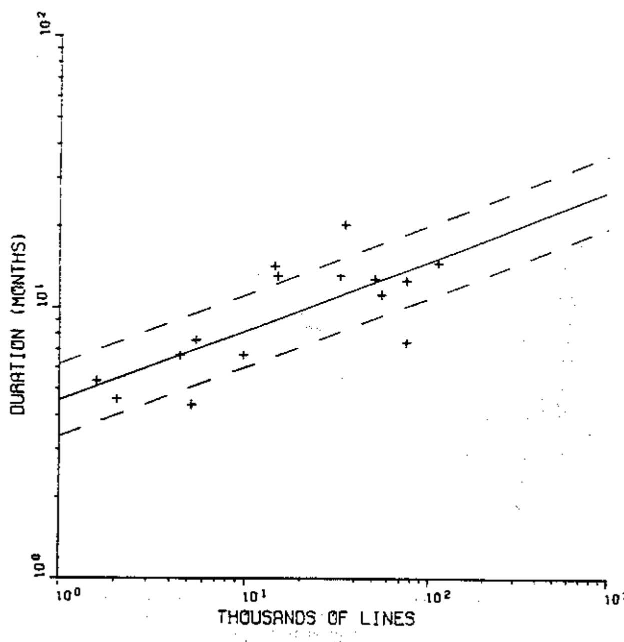


Figure 8. Duration vs lines of code: $D = 4.55*(L**0.26)$.

and the relationship between duration and the number of modules is shown in Figure 9. The equations for these relationships are

$D = 4.55*(L**0.26)$, (19)
$D = 1.96*(M**0.33)$, (20)

respectively. Walston and Felix found a nearly cubic relationship between project duration and delivered code:

$D = 4.1*(L**0.365)$. (21)

This relationship is quite similar to that found for the SEL data (see Figure 10).

Reusing code or modules may have an effect on project duration. The relationships between project duration and new lines of code in thousands, developed lines in thousands, and new modules are

$D = 4.62*(NL**0.28)$, (22)
$D = 4.58*(DL**0.28)$, (23)
$D = 2.5*(NM**0.30)$, (24)

respectively. These relationships are very close to (19) and (20). As one might expect, calendar time increases at about one-third the rate of size. This is owing to the fact that calendar time on larger projects is a major constraint and more people are required to meet the calendar deadlines.

Project duration as a function of total effort is shown in Figure 11. The regression equation is

$D = 4.39*(E**0.26)$. (25)

Walston and Felix also found that a cubic relationship exists between project duration and total effort:

$D = 2.47*(E**0.35)$. (26)

Equations (25) and (26) are very similar. A comparison of the two estimating equations is shown in Figure 12. The SEL equation lies about one standard error of es-

Figure 9. Duration vs number of modules: $D = 1.96*(M**0.33)$.

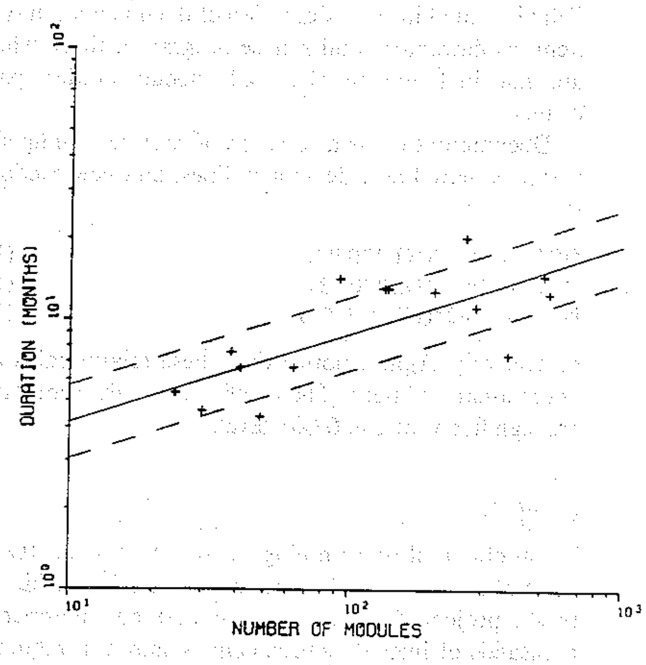
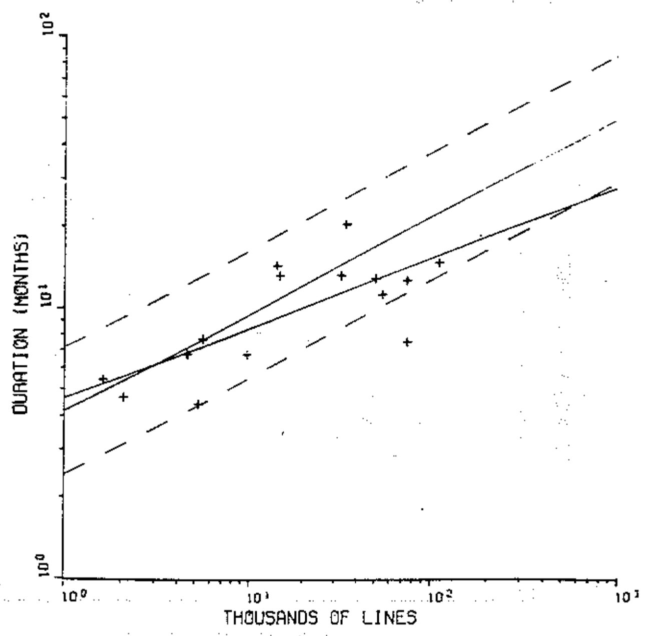


Figure 10. Duration vs lines of code.



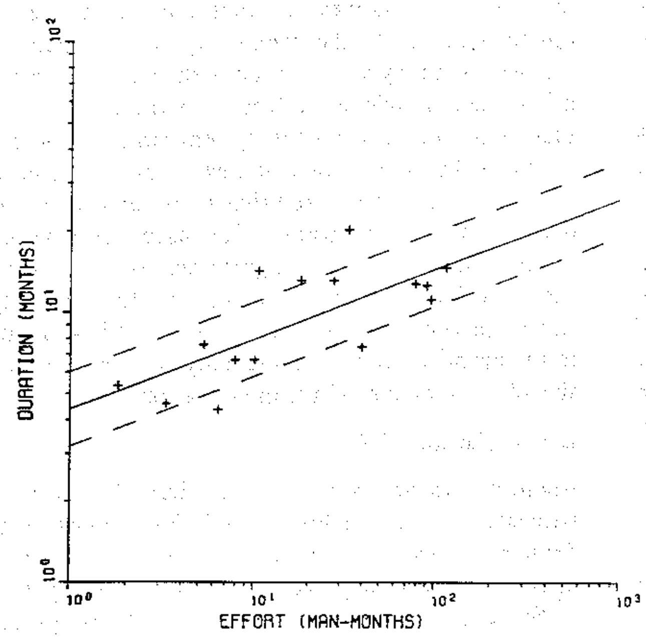


Figure 11. Duration vs effort: $D = 4.39*(E**0.26)$.

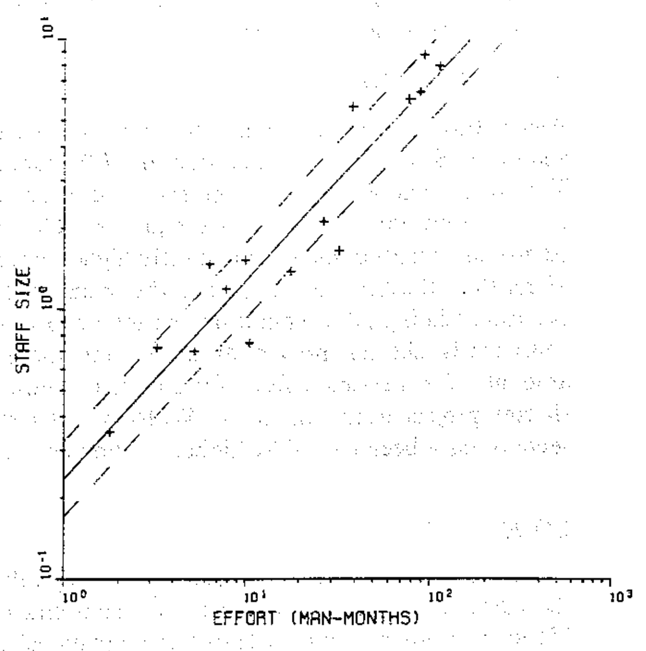


Figure 13. Staff size vs effort: $S = 0.24*(E**0.73)$.

timate above the Walston–Felix equation. More will be said about this relationship in the next section.

Staff Size

The staff size used for the development of a software product depends on several factors, including the development time allowed for the project, the amount and difficulty of the code to be produced, and the manpower loading rates that can be achieved [10]. The equation relating average staff size (total man months of effort divided by the project duration in months) and total effort (Figure 13) is

$$S = 0.24*(E**0.73). \tag{27}$$

The coefficient of determination indicates a good rela-

Figure 12. Duration vs effort.

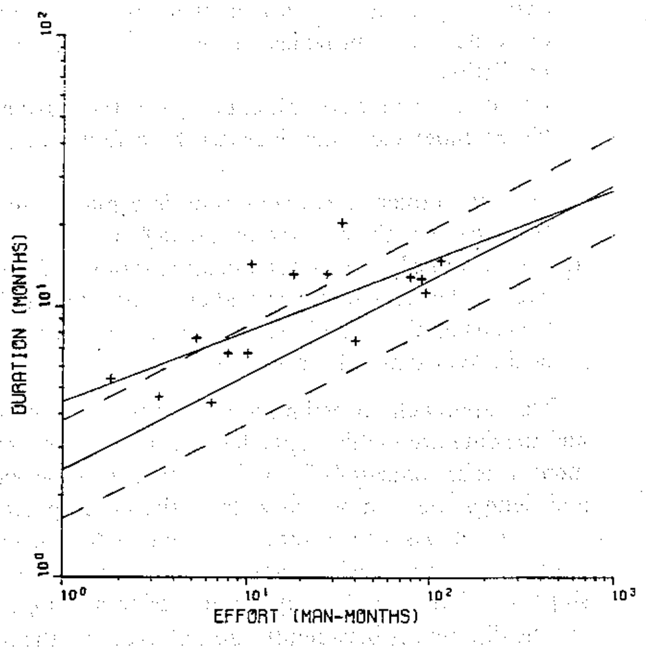
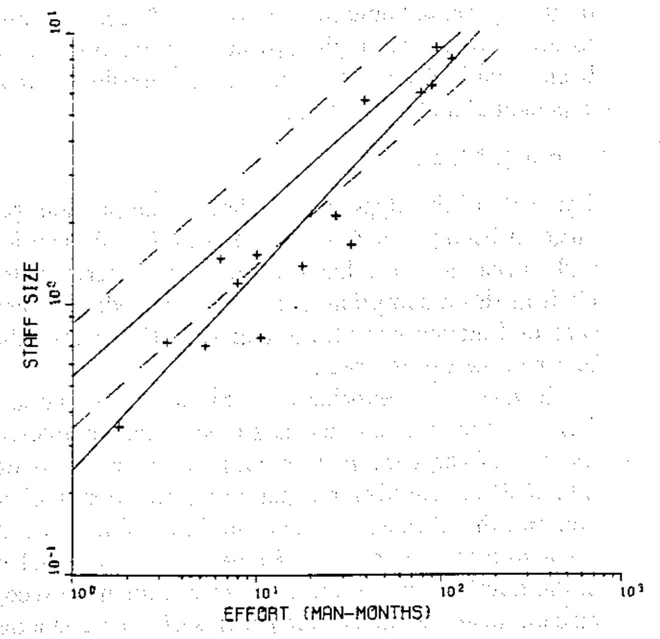


Figure 14. Staff size vs effort.



tionship between these two variables. The Walston–Felix equation for this relationship is

$$S = 0.54*(E**0.6). \tag{28}$$

Again, this equation is very much like (27). Figure 14 shows the SEL estimating equation and (28) together. The SEL equation lies about one standard error of estimate below the Walston–Felix equation. The Walston–Felix equation shows consistently higher manning levels than the SEL equation, but the Walston–Felix equation relating project duration and total effort shows consistently shorter project durations for the same amount of expended effort (Figure 12). Thus, the shorter project durations in the Walston–Felix study seem to have been gained by higher staffing levels.

CONCLUSIONS

The authors believe they have been able to help validate the basic relationships reported by Walston and Felix [5] in their original study. Clearly, the equations' coefficients are different for different environments, as one would expect, but there is a consistency in the way the SEL equations relate to the Walston–Felix equations.

On the other hand, the SEL data could also be used to support the linear relationship between effort and lines of code described by Jeffery and Lawrence [4]. Their data deal with business applications, predominantly in COBOL, ranging in size from 100 to 4500 lines of code. Their effort includes detailed design, coding, and testing. The SEL data lies between the Jeffery–Lawrence data and the Walston–Felix data with respect to size.

Whether the relationship between effort and lines of code is modeled by a linear equation ($E = a*L + b$) or an exponential equation $E = a*(L**b)$, it is closer to linear than one might expect. For example, it has been hypothesized [11] that the relationship is more exponential and of the form

$$E = a*(L**1.5).$$

The basis of this hypothesis is that as the problem gets larger it becomes more difficult to develop the solution, and so the effort per line of code should increase. Implicit in this assumption is that lines of code is a measure of function complexity and that the relationship between the two is linear.

However, it is possible that this last assumption is false. As the problem increases in size and complexity, the size of the code may increase at an even greater rate. This increased rate is due to subfunction duplication and the looseness of the code. For example, as the problem increases in size and more people are involved in the development, it becomes more difficult to recognize duplicate function. Much of this duplicate function may be simple routines that each programmer redevelops for himself. As the complexity of the function increases, as it may very well do with size, there may be a looseness of code, a tendency to write a longer, simpler algorithm to keep the system simple. There are limits to the amount of complexity an individual can handle. It is also often true that there tends to be more overdesign of subprograms. The insecurity caused by the pure size forces the programmer to overdesign for safety, which results in more code per function. All of this extra code creates a larger system whose relation to the problem grows exponentially with respect to the size of the problem. Thus, the equation

$$E = a*(\text{function}**b),$$

where b is about 1.5, may be true, but when compared with size measured as lines of code b is closer to 1. Unfortunately, we are unable to measure function and complexity accurately enough to verify this hypothesis.

Some comments on the basic relationships seem worth making:

a. Based on the SEL data, it appears that developed lines and new lines are a better estimate of effort than total lines. This is intuitively satisfying since it is closer to the notion of expended effort.
b. Even though the measure of productivity is rather primitive, there is a tendency to believe from our data that reusing code is cost effective. Because of different ways of counting reused code, we were unable to compare our data with that of Walston and Felix.
c. The use of modules as a measure of effort works about as well for the SEL environment as various measures of lines of code. Since in many cases it is easier to predict the number of modules than lines of code, this provides a viable approach to prediction.
d. Productivity in environments where the design is better understood may increase by a factor of 3 or 4.
e. On large projects, calendar time is a major factor. It increases with the cube root of effort.
f. The relationships between documentation and product size, between duration and effort or size, and between staff size and effort are reasonably supportive of the Walston–Felix relationships.

This approach to estimation is empirically based, and the data are highly dependent on the local environment. It is an indicator of how we currently do business and defines the common aspects of the developmental environment. As new projects are added to the data base, the equations will change and the base relationships will change as the way we do business changes.

The differences between the actual data and the predicted values of the equations can be explained by variations in the environmental factors for the different projects within the SEL, including methodology and constraints. We can think of the basic lines of code and effort as capturing the essential SEL environment and the individual projects as requiring modification due to specific variations within the project environment. This approach was used by Walston and Felix in their productivity index and by Boehm ([12]) in his COCOMO model. We are currently investigating this approach by developing a metamodel that will be adapted to the local organizational and project environment by isolating local SEL environmental factors.

ACKNOWLEDGMENTS

The authors are grateful to Frank McGarry of NASA/Goddard Space Flight Center, Jerry Page and Victor Church of Computer Sciences Corporation, Claude Walston and Charles Felix of IBM Federal Systems Division, and John Bailey of the University of Maryland for their comments on drafts of this paper and for supplying data needed for this paper.

REFERENCES

1. V. R. Basili, M. V. Zelkowitz, F. E. McGarry, R. W. Reiter, W. F. Truszkowski, and D. L. Weiss, *The Software Engineering Laboratory,* Tech. Rep. TR-535, Department of Computer Science, University of Maryland, May 1977.
2. V. R. Basili and M. V. Zelkowitz, Analyzing Medium Scale Software Development, *Proceedings of the Third International Conference on Software Engineering,* Atlanta, Georgia, May 1978, pp. 116–132.
3. K. Freburger and V. R. Basili, *The Software Engineering Laboratory: Relationship Equations,* Tech. Rep. TR-764, Department of Computer Science, University of Maryland, May 1979.
4. D. R. Jeffery and M. J. Lawrence, *An Inter-organizational Comparison of Programming Productivity,* Department of Information Systems, University of New South Wales, 1979.
5. C. E. Walston and C. P. Felix, A method of Programming Measurement and Estimation, *IBM Syst. J.* 16, 1 (1977).
6. E. Chrysler, Some Basic Determinants of Computer Programming Productivity, *Commun. ACM* 21, 6 (1978).
7. J. R. Johnson, A Working Measure of Productivity, *Datamation* 23, 2 (1977).
8. V. LaBolle, *Development of Equations for Estimating the Costs of Computer Program Production,* System Development Corporation, Santa Monica, California, 1966.
9. E. A. Nelson, *Management Handbook for the Estimation of Computer Programming Costs,* System Development Corporation, Santa Monica, California, 1967.
10. L. H. Putnam, A General Empirical Solution to the Macro Software Sizing and Estimating Problem, *IEEE Trans. Software Eng.* SE-4, 345–361 (1978).
11. F. P. Brooks, *The Mythical Man-Month,* Addison-Wesley, Reading, Massachusetts, 1975.
12. B. W. Boehm, *Software Engineering Economics,* Prentice-Hall, Englewood Cliffs, New Jersey, 1981.