A META-MODEL FOR SOFTWARE DEVELOPMENT RESOURCE EXPENDITURES

John W. Bailey and Victor R. Basili

Department of Computer Science
University of Maryland, College Park 20742

ABSTRACT

One of the basic goals of software engineering is the establishment of useful models and equations to predict the cost of any given programming project. Many models have been proposed over the last several years, but, because of differences in the data collected, types of projects and environmental factors among software development sites, these models are not transportable and are only valid within the organization where they were developed. This result seems reasonable when one considers that a model developed at a certain environment will only be able to capture the impact of the factors which have a variable effect within that environment. Those factors which are constant at that environment, and therefore do not cause variations in the productivity among projects produced there, may have different or variable effects at another environment.

This paper presents a model-generation process which permits the development of a resource estimation model for any particular organization. The model is based on data collected by that organization which captures its particular environmental factors and the differences among its particular projects. The process provides the capability of producing a model tailored to the organization which can be expected to be more effective than any model originally developed for another environment. It is demonstrated here using data collected from the Software Engineering Laboratory at the NASA/Goddard Space Flight Center.

INTRODUCTION

Several resource estimation models for a software-producing environment have been reported in the literature [1,2,3,4,5,6,7,8,9], each having been developed in a different environment, each having its particular strengths and weaknesses but with most showing fairly poor characteristics concerning portability to other environments. It is becoming apparent that it is not generally possible for one software development environment to use the algorithms developed at another environment to predict resource consumption. It is necessary for each environment to consider its own past productivity in order to estimate its future productivities. Traditionally, a good manager can estimate resource consumption for a programming project based on his past experience with that particular environment. A model should be able to do the same, and can serve as a useful aid to the manager in this estimating task.

However, if a manager uses a model developed at another environment to help him in his estimations, he will usually find that his intuitive estimates are better than any from the model. It would be advantageous for his software-development organization to generate a model of its own by duplicating the basic steps taken in the development of some outside environment's estimation model. The organization could parallel its own model's development with the development of the existing model, making decisions along the way with respect to which factors have an effect on its software environment, and could mold the newly emerging model to its specific environment. This is seen as an additional advantage over those models which are only "tuned" to the user's environment via a set of specified parameters, since in the latter case there may be no way to express certain peculiarities of the new environment in terms which the model can handle. When one considers in general how poorly a model from one environment fits another environment, it seems that such peculiarities are the rule rather than the exception. Unfortunately, there have been few attempts to reveal the steps taken in generating a resource estimation model which would be helpful to any organization wishing to establish a model for its own use.

This paper is a first attempt by the Software Engineering Laboratory of the University of Maryland at College Park to outline the initial procedures which we have used to establish this type of model for our environment. It is hoped that the framework for the model presented here is general enough to help another software development organization produce a model of its own by following a similar procedure while making decisions which mold the model to its own environment.

One basic approach will be outlined and developed here, but several variations will be discussed. The type of model used is based on earlier work of Walston and Felix at IBM Federal Systems Division and Barry Boehm at TRW in that it attempts to relate project size to effort. Some reasonable measure is used to express the size of a project, such as lines of source code, executable statements, machine instructions or number of modules, and, a base-line equation is used to relate this size to effort. Then, the deviations of the actual projects from this prediction line are explained by some set of factors which attempt to describe the differences among projects in the environment. These factors may include measures of skill and experience of the programming team, use of good programming practices and difficulty of the project.

Several of the alternatives became apparent during our study and these are mentioned when appropriate even if they are not examined further here. Although some of the details and ideas used in this study may not pertain to other environments, it is hoped that enough possibilities are given to show the general idea of how the technique we used can be applied. The study now involves complete data on eighteen projects and sub-projects but was begun when we had complete data on only five projects. It is hoped that the presentation of our work will save other investigators who are developing a model some time or at least provide a point of departure for their own study.

Background

There exist many cost estimation models ranging from highly theoretical ones, such as Putnam's model [1], to empirical ones, such as the Walston and Felix [2] and the Boehm model [3]. An empirical model uses data from previous projects to evaluate the current project and derives the basic formulae from analysis of the particular data base available. A theoretical model, on the other hand, uses formulae based upon global assumptions, such as the rate at which people solve problems, the number of problems available for solution at a given point in time, etc. The work in this paper is empirical and is based predominantly on the work of Walston and Felix, and Barry Boehm.

The Software Engineering Laboratory (SEL) has worked to validate some of the basic relationships proposed by Walston and Felix which dealt with the factors that affect the software development process. One result of their study was an index computed with twenty-nine factors they judged to have a significant effect on their software development environment. As part of their study, they proposed an effort equation which was of the form $E = 5.2 * L^{.91}$ where E is the total effort in man-months and L is the size in thousands of lines of delivered source code. Data from SEL was used to show that although the exact equation proposed by Walston and Felix could not be derived, the basic relationship between lines of code and effort could be substantiated by an equation which lay within one standard error of estimate for the IBM equation, and in a justifiable direction [10]. Barry Boehm has proposed a model that uses a similar standard effort equation and adjusts the initial estimates by a set of sixteen multipliers which are selected according to values assigned to their corresponding attributes. In attempting to fit an early version of this model, but with the SEL data, it was found that because of differing environments, a different baseline equation was needed, as well as a different set of environmental parameters or attributes. Many of the attributes found in the TRW environment are already accounted for in the SEL baseline equations, and several of the attributes in the SEL model which accounted for changes in productivity were not accounted for in the Boehm model, presumably because they had little effect in the TRW environment. Based upon this assumption and our experience with the IBM and TRW models, the meta model proposed in this paper was devised.

The SEL Environment

The Software Engineering Laboratory was organized in August, 1976. Beginning in November, 1976, most new software tasks that were assigned by the System Development Section of NASA/Goddard Space Flight Center began submitting data on development progress to our data base. These programs are mostly ground support routines for various spacecraft projects. This usually consists of attitude orbit determinations, telemetry decommutation and other control functions. The software that is produced generally takes from six months to two years to produce, is written by two to ten programmers most of whom are working on several such projects simultaneously, and requires from six man-months to ten man-years of effort. Projects are supervised by NASA/GSFC employees and personnel are either NASA personnel or outside contractors (Computer Sciences Corporation).

The development facility consists of two primary hardware systems: a pair of S/360's and a PDP-11/70. During development of software systems users can expect turn-around time to vary from one or two hours for small, half-minute jobs, to one day for medium jobs (3 to 5 minutes, less that 600K), to several days for longer and larger jobs. The primary language used is FORTRAN although there is some application of assembler language.

THE META-MODEL

The meta-model described here is of the adjusted base-line type such as those proposed by Walston and Felix and Barry Boehm. Therefore, the basic approach is a two-step process. First, the effort expended for the average project is expressed as a function of some measure of size and, second, each project's deviation from this average is explained through the systematic use of a set of environmental attributes known for each project. The remainder of this paper will describe this process and will follow the format:

1) Compute the background equation
2) Analyze the factors available to explain the difference between actual effort and effort as predicted by the background equation
3) Use this model to predict the effort for the new project

The Background Equation

The background or base-line relationship between effort and size forms the basis for the local model. It is found by fitting some choice of curve through the scatter plot of effort versus size data. By definition, then, it should be able to predict the effort required to complete an average project, given its size. This average effort value as a function of size alone has been termed the "standard effort" throughout this paper. This section deals with:

1.1) Picking and defining measures of size and effort
1.2) Selecting the form of the base-line equation
1.3) Calculating an initial base-line for use in the model

In any given environment the decision of what size measure to use would have to depend initially upon what data is available. In our case, it was decided that size could be measured easily by lines of source code or by modules and that effort could be expressed in man-months. Consideration should also be given to the ease with which each measure can be estimated when the model is used to predict the effort required for future projects. The upper management in our programming environment was of the opinion that source lines with comments was the easier of the two readily available measures to predict. Also, it was decided that, based upon the data available and the ultimate use of the model, project effort would be defined to be measured from the beginning of the design phase through acceptance testing and to include programming, management and support hours.

In our data base, the total number of lines and modules as well as the number of new lines and new modules were available for the 18 projects and sub-projects. Initially, we expressed effort in terms of each of the four size measures mentioned above. To do this, we used three forms of equations to fit the data, using both the raw data and logarithms of the data, which provided functions we hoped would express the basic relationship between size and effort that exists in our environment. The forms of the three types of equations were:

E = effort S = size

$$E = a * S + b \quad (1)$$

$$E = a * S^b \quad (2)$$

$$E = a * S^b + c \quad (3)$$

Some difficulties were encountered when attempting to fit a conventional least-squares regression line through the raw data. One probable reason for this is that a correlation between the deviations from the prediction line and the size of the project could not easily be eliminated (heteroscedasticity). Rather than using a least-squares line with a single, arithmetic standard error of estimate which would be consistently large with respect to small projects and often too small when applying the equation to large projects, we opted for a prediction line which minimized the ratio between the predicted values for effort and each actual data point. In this way, the standard error is multiplicative and can be thought of as a percent error whose absolute magnitude increases as the project size increases. If, however, equations of the second or third form are derived by fitting a least-squares line through the logarithms of the data, the standard error automatically becomes multiplicative when converted back to linear coordinates.

The third form shown above was the most successful for us. It was in the form of an exponential fit but included a constant which removed the constraint that the prediction line pass through the origin. This line was not found by converting to logarithms but by an algorithm that selected the values which minimized the standard error of estimate when expressed as a ratio. The theory behind the implementation of this multiplicative standard error of estimate is described later. Although the size of our data base was not large enough to firmly support using this fit rather than a straight line, we are using it here primarily as an illustration, and therefore felt justified in retaining it.

Turning back to the measurement of size, it was noted that neither the equations based upon size in terms of new lines of code or new modules nor those based upon total lines of code or total modules captured the intuitive sense of the amount of work required for each project. It was felt that although using previously-written code was easier than generating new code, the integration effort was still significant and should be accounted for. After examining the background relationships discussed above, another more satisfying measurement for size was derived. Instead of considering only the total lines or only the new lines to determine the size of a project, an algorithm to combine these sizes into one measure was selected. It was found that by computing the effective size in lines to be equal to the total number of new lines written plus 20% of any old lines used in the project, a base-line relationship of lower standard error could be derived. This new size measure will be called "developed lines" in this paper. The same technique was applied to numbers of modules and resulted in a measure of "developed modules." Other proportions of new and old sizes were tried as well as an algorithm which computed developed size based on a graduated mixture of new and old code where larger projects counted a higher percentage of their re-used code in the developed size. Often, these equations did produce slightly better background relationships, but the improvement in standard error was judged not to be worth the added complexity. It was hoped that as long as some reasonable algorithm was selected which captured the size as measured by both the amount of new product as well as old product, most of the remaining differences among the projects should be explainable by the varying environmental attributes.

At this point, the three base-line equations, based on the computed sizes of developed lines only, were:

E = effort in man-months of programming and management time
DL = number of developed lines of source code with comments (new lines with comments plus 20% of re-used lines)

Equation:	*Standard error of estimate:	
$E = 1.36*DL + 1.62$	1.269	(4)
$E = 1.86*DL^{.93}$	1.297	(5)
$E = 0.73*DL^{1.16} + 3.5$	1.250	(6)

* Note that these are multiplicative factors. The predicted value given by the equation is multiplied and divided by this factor to get the range for one standard error of estimate. All standard errors of estimate (s.e.e.) in this paper are of this type.

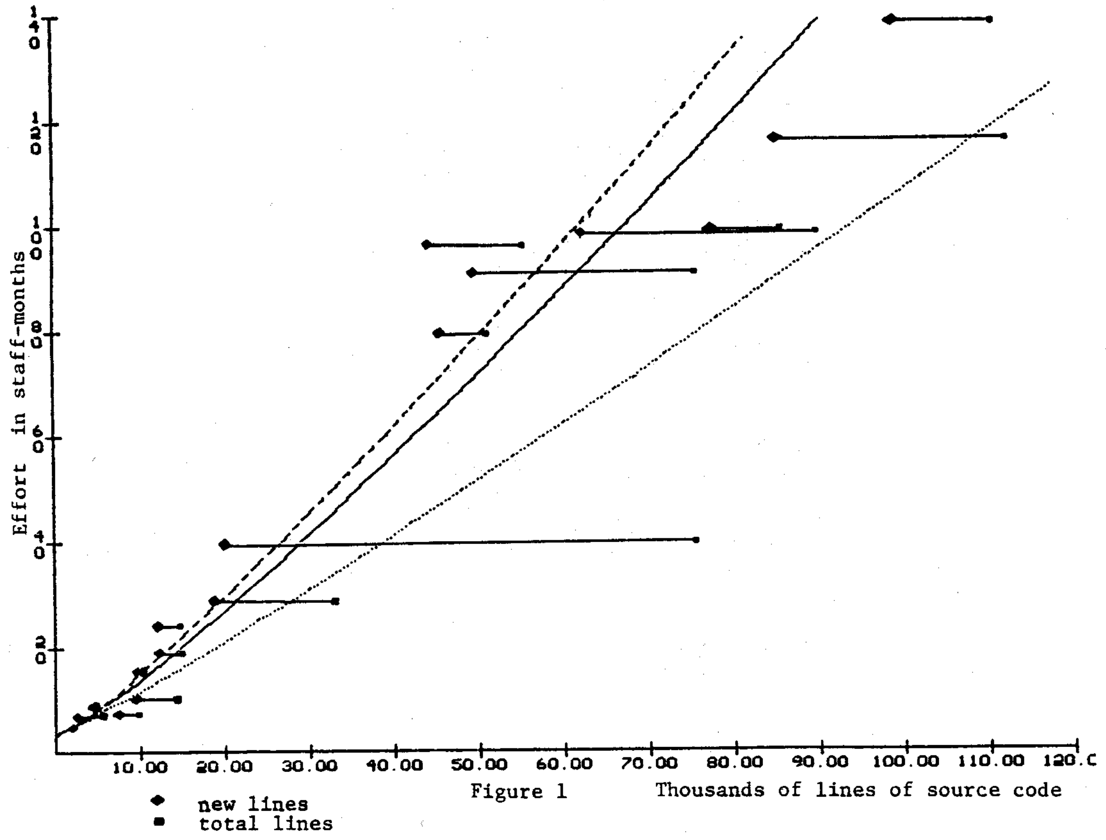

Figure 1

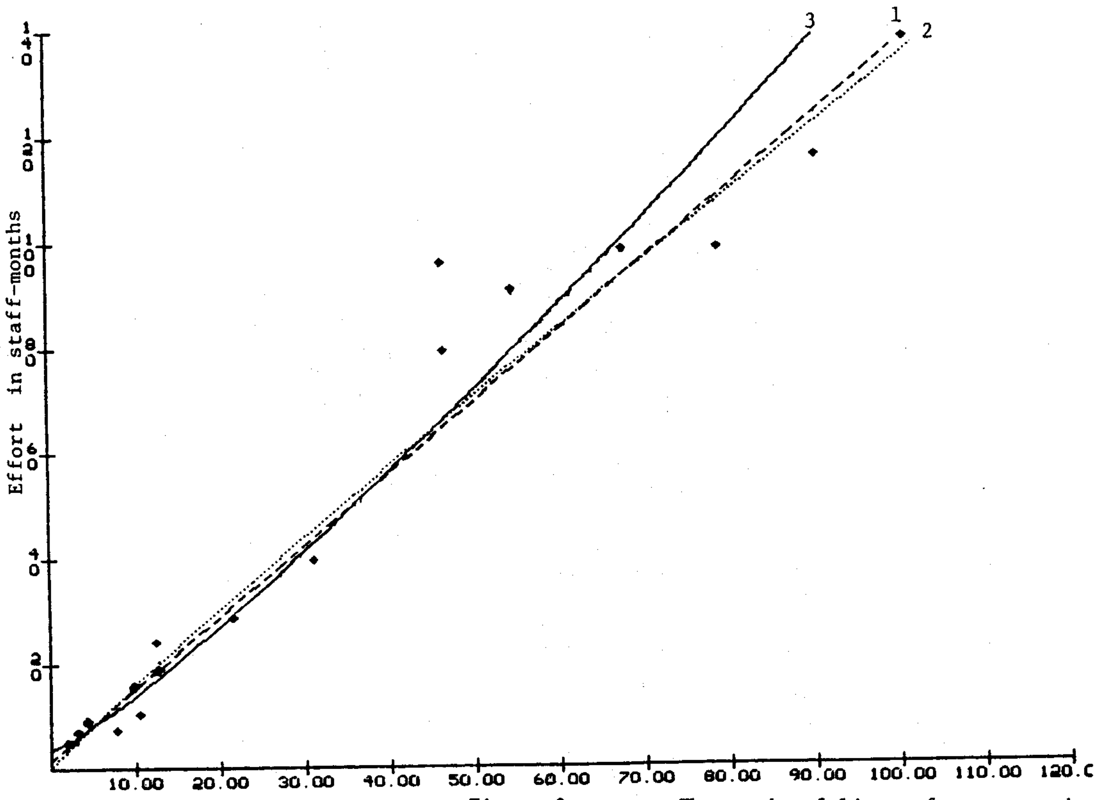

Figure 2

Figure 1 shows how the exponential fit with constant for developed lines falls between those for new lines and total lines, hopefully doing a better job than either of the other two in relating a project's size to the resources consumed during its development. The remainder of this paper will deal entirely with this computed measure of size since it was our most successful expression for work output for a given project.

Figure 2 shows these three background prediction equations superimposed on the data points. It was decided to use equation 3, above, as the base-line throughout the remainder of the model generation since it achieved the best fit to the data points and suggested the intuitively satisfying fact that a project requires a minimum overhead effort (the Y-intercept of the function). Equation one, a straight line, does as well statistically, and could well have been adopted for simplicity. Since this is meant to be an illustration, however, and it was felt that the non-linear relationship between size and effort was more common outside of our environment, equation three was adopted for use in this study. The remaining errors of estimation appear as the vertical distances between each point and the line. It is these distances in the form of ratios which we would like to explain in terms of the environmental attributes.

Project Factors

The next step in determining a model is to collect data about the programming environment of each project which captures the probable reasons why some projects took more effort and thereby consumed more resources than others when normalized for size. This data could include such factors as methodologies used during design and development, experience of the customer and of the programmers, managerial control during development, number of changes imposed during the development and type and complexity of the project. It is assumed that the correct application of information such as this can assist in explaining the variations observed among projects in terms of their productivities. The steps described in this section include:

2.1) Choosing a set of factors
2.2) Grouping and compressing this data
2.3) Isolating the important factors and groups
2.4) Incorporating the factors by performing a multiple regression to predict the deviations of the points from the computed base-line

In all, close to one hundred environmental attributes were examined as possible contributors to the variations among the productivities of the projects. Table 1 shows a list of these factors as well as some others which we did not use. Thirty-six of the factors were those used by Walston and Felix, sixteen were used by Boehm and 30 others were suggested by our environment. Although we did not use all these factors, they are included to provide additional ideas for other investigators. It should be noted that it is not necessary to consider any factors which are constant for the set of projects currently in the data-base since the influence of this factor will already be contained in the base-line relationship. If, however, a future project is rated differently in one of these categories, it may be necessary to reinstate it into the model.

The process of selecting attributes to use is largely a matter of what information is available. Since many of the projects we studied were completed when this investigation began, it was necessary to rely on project management for the information required. The inclusion of past projects was justified in order to establish as large a data-base as possible, however, it made it necessary to be particularly careful about the consistency between the ratings for current projects and those for projects already completed. To maintain the integrity of the values of these attributes, all ratings produced by the vendor's management were examined by the customer's management and also by us. In this way we hoped to avoid the temptation to adjust ratings to reflect the known ultimate success of past projects.

Many of the attributes required no special work to assign a value, such as "Team Size" or "Percent Code: I/O," but most required imposing a scale of some kind. We decided that an exact scale was not possible or even necessary so a six-point subjective rating was used. This format was chosen by the managers who would be making the ratings since it conformed well with the information they had already collected about many of the attributes. Most of the factors, then, are rated on a scale from 0 to 5 with 5 being the most of that particular attribute (whether it is "good" or "bad"). The most important point is that we tried to remain consistent in our ratings from project to project. The need for this was particularly noticeable when rating earlier projects in terms of development methodology. For instance, what may have been thought of as a "4" rating in "Formal Training" for a project which began coding over a year ago may actually be a "3" or even a "2" when compared with the increased sophistication of more recent projects. We found it necessary to re-scale a few of the attributes because of this consideration.

After a set of environmental factors is selected and the data collected, it is necessary to consider the number of these attributes versus the number of projects in the data base. It is not statistically sound to use a large group of factors to predict a variable with relatively few data points. Unless a very large number of projects is being used, it will probably be necessary to condense the information contained in the whole set of factors into just a few new factors. This can be accomplished entirely intuitively, based on experience, or with the help of a correlation matrix or factor analysis routines. Although there is no absolute rule as to how many factors should be used to predict a given number of points, a rule of thumb might be to allow up to ten or fifteen percent of the number of data points. Strictly speaking, the adjusted r-squared values or the F-values should be observed as factors are added to the prediction equation via a multiple regression routine (described below) to avoid the mistake of using too many factors.

In our environment, we had data on 71 attributes which we suspected could affect the ultimate

Walston and Felix:

- Customer experience
- Customer participation in definition
- Customer interface complexity
- Development location
- Percent programmers in design
- Programmer qualifications
- Programmer experience with machine
- Programmer experience with language
- Programmer experience with application
- Worked together on same type of problem
- Customer originated program design changes
- Hardware under development
- Development environment closed
- Development environment open with request
- Development environment open
- Development environment RJE
- Development environment TSO
- Percent code structured
- Percent code used code review
- Percent code used top-down
- Percent code by chief-programmer teams
- Complexity of application processing
- Complexity of p.ogram flow
- Complexity of internal communication
- Complexity of external communication
- Complexity of data-base structure
- Percent code non-math and I/O
- Percent code math and computational
- Percent code CPU and I/O control
- Percent code fallback and recovery
- Percent code other
- Proportion code real time of interactive
- Design constraints: main storage
- Design constraints: timing
- Design constraints: I/O capability
- Unclassified

Boehm:

- Required fault freedom
- Data base size
- Product complexity
- Adaptation from existing software
- Execution time constraint
- Main storage constraint
- Virtual machine volatility
- Computer response time
- Analyst capability
- Applications experience
- Programmer Capability
- Virtual machine experience
- Programming language experience
- Modern programming practices
- Use of software tools
- Required development Schedule

SEL:

- Program design language (development and design)
- Formal design review
- Tree charts
- Design formalisms
- Design/decision notes
- Walk-through: design
- Walk-through: code
- Code reading
- Top-down design
- Top-down code
- Structured code
- Librarian
- Chief Programmer Teams
- Formal Training
- Formal test plans
- Unit development folders
- Formal documentation
- Heavy management involvement and control
- Iterative enhancement
- Individual decisions
- Timely specs and no changes
- Team size
- On schedule
- TSO development
- Overall
- Reusable code
- Percent programmer effort
- Percent management effort
- Amount documentation
- Staff size

Table 1

productivity of a project, but only 18 projects for which to see the results. We found it necessary, therefore, to perform such a compression of the data. Our next step, then, was to examine the attributes and group into categories those which we felt would have a similar effect on the project. As an aid to selecting potential groupings for analysis, a correlation matrix for all the attributes was studied. It was hoped that meaningful groups could be formed which would retain an intuitive sense of positive or negative contribution to the project's productivity. By studying the potential categorizations of the factors, and how they performed in potential models to predict developed lines, we settled upon three groups using 21 of the original attributes. The groups and their constituent attributes were:

Total Methodology
- Tree Charts
- Top Down Design
- Design Formalisms
- Formal Documentation
- Code Reading
- Chief Programmer Teams
- Formal Test Plans
- Unit Development Folders
- Formal Training

Cumulative Complexity
- Customer Interface Complexity
- Customer-Initiated Design Changes
- Application Process Complexity
- Program Flow Complexity
- Internal Communication Complexity
- External Communication Complexity
- Data Base Complexity

Cumulative Experience
- Programmer Qualifications
- Programmer Experience with Machine
- Programmer Experience with Language
- Programmer Experience with Application
- Team Previously Worked Together

We were particularly interested in using a methodology category due to the findings of Basili and Reiter [11] which implied improvement in the development process due to the use of a specific discipline. The methodology category was selected to closely coincide with the principles of the methodology used in the experiment. The complexity category was included to account for some of the known negative influences on productivity. The cumulative rating for each of these categories was merely a sum of the ratings of its constituents (each adjusted to a 0 to 5 scale). Although it was necessary to reduce the number of attributes used in the statistical investigation in this manner in order to give more meaningful results, the simple summing of various attributes loses some of the information which could be reflected in these categories. This is because even though one of the constituent attributes may be much more important than another, an unweighted sum will destroy this difference. One solution to this type of dilemma is to have many more data points, as mentioned before, and to use the attributes independently. Another would be to determine the relative effects of each attribute and to weight them accordingly. Without the necessary criteria for either of these solutions, however, we were forced to continue in this direction and to accept this trade-off.

Incorporating the Factors

The purpose of the attribute analysis is to explain the deviations displayed by each project from the derived background equation and, ultimately, to yield a prediction process where the attributes can be used to determine how far a project will "miss" the background equation, if at all.

The next step, then, is to compute these differences which must be predicted. A quantity based on the ratio between the actual effort expended and the amount predicted by the background equation was used as a target for the prediction. In this way, when the model is in use, the background equation can be applied to determine the standard effort (the amount needed if the project behaved as an average of the previous projects in the data-base). Then, the attributes will be used to yield a ratio between this rough estimate and a hopefully more accurate expected value of the effort required.

The SPSS [12] forward multiple regression routine was used to generate an equation which could best predict each of the project's ratio of error. The actual ratio was converted to a linear scale with zero meaning the actual data point fell on the base line. This was accomplished by subtracting one from all ratios greater than one and adding one to the negative reciprocals of those ratios which were less than one. For instance, if a project's standard effort was predicted to be 100 man-months and it actually required 150 man-months, this ratio would be 1.5. Subtracting one makes this project's target value 0.5. If however it had needed only 66.7 man-months, its ratio would be .667 which is less than one. Adding one to the negative reciprocal of this number gives a target value of -0.5. The assumption is that this scale tends to be symmetrical in that the first project had as many negative factors impact its productivity as the second project had positive.

In the first pass at using the multiple regression routine, we were using five attribute groups. Since the data base was not very large, we were cautious about assigning any useful significance to the results. We therefore recondensed the attribute data into the three groups shown above. The results of this attempt are described in a later section.

Variations on the Model

We noticed that it was possible to combine the two processes of first isolating a background equation and then applying the environmental attributes to explain deviations from that equation into a single procedure. To do this, a measure of size was included as a factor with the set of environmental attributes and the whole group was used to predict effort. As expected, size was always chosen first by the forward routine, since it correlated the best with effort for each project. This single process lacked the intuitively satisfying intermediate stage which related to a base-line relationship as a half-way point in the model's results, but it streamlined the model somewhat.

In order to preserve the possibility of an exponential relationship between size and effort, this method was used with the logarithms of the size and effort values. The output of the regression analysis would be of the form,

$$\log(\text{Effort}) = A*\log(\text{Size}) + B*\text{attr1} + C*\text{attr2} + \ldots + K \qquad (7)$$

This would convert to,

$$\text{Effort} = \text{Size}^{A} * 10^{(B*\text{attr1}+C*\text{attr2}+\ldots+K)} \qquad (8)$$

assuming, here, that log base 10 was used in the conversion.

A third template for a model was tried which attempted to eliminate nearly all of the reliance on the actual numerical values of our attribute ratings in order to legitimize some of our statistical analyses. Only two of the attribute groups mentioned before were considered, "Complexity" and "Methodology." Each of these two ratings were transformed into two new ratings of binary values resulting in four new attributes, "High Methodology," "Low Methodology," "High Complexity," and "Low Complexity." The transformation was accomplished as follows: if a project's rating fell in the upper third of all projects, the value of the "High" binary attribute of that type was assigned a 1 while the value of the "Low" attribute for that type was assigned a 0. If the value fell in the middle third, both binary values were assigned a 0. If the value fell in the low third, the "Low" attribute was assigned a 1 while the "High" was assigned a 0. This reduced our assumptions about the data to the lowest level for statistical analysis. For illustration, call the four new binary attributes HM, LM, HC, LC for high and low methodology and high and low complexity. The result of the multiple regression analysis, then, would be in the form,

$$\text{Effort} = \text{Size}^{A} * 10^{(B*HM+C*LM+D*HC+E*LC+K)} \qquad (9)$$

Since the chance that any chosen attribute value will be 0 for a particular project is about 2/3, most of those terms on the right will drop out when the model is actually applied to a given project. Although we did not expect to achieve the same accuracy from this method, the simplicity of it was appealing.

APPLYING THE MODEL

As an illustration of the results obtained thus far for our environment, this section deals with the actual values of the data we used and the models we generated. It should serve as a useful guide and a summary of the steps we chose to follow. In order to include an illustration of the functioning of the completed model, one project, the most recently completed project, will be removed from the analysis while a new model is developed. This project will then be treated as a new data point in order to test and illustrate the performance of the model. Typically, the use of the model will involve the following steps:

3.1) Estimate size of new project
3.2) Use base-line to get standard effort
3.3) Estimate necessary factor values
3.4) Compute difference this project should exhibit
3.5) Apply that difference to standard effort

Appendix 1 shows the eighteen projects and sub-projects currently in our data-base with the measures of size previously discussed. As stated above, developed size is all of the newly-written lines or modules plus 20% of the re-used lines or modules, depending on which size measure is being used. The developed size is what we chose to predict with the models generated. We also chose, as a baseline, the exponential equation with the constant term. The following illustration shows the development of the model with the first seventeen points in the data base. The base-line relationship between developed lines of code and effort was:

$$E = .72 * DL^{1.17} + 3.4 \quad (\text{s.e.e.}=1.25) \qquad (10)$$

The remaining information used about the projects is shown in the appendix. The remaining error ratios from this line to each project's actual effort were computed and listed. These are the values which should be explained by the multiple regression analysis. When the model is in use, then, an error ratio can be derived by using the multiple regression equation which can then be applied to the base-line equation to provide what should be an even better estimate of effort than the base-line alone. As discussed, the three main categories of environmental attributes shown are the result of distilling many attributes.

The equations computed by the SPSS forward multiple regression routine which attempt to express the list of error ratios as functions of various of the attributes provided are:

ER = Effort ratio (converted to linear scale)
METH = Methodology
CMPLX = Complexity

$$ER = -.036 * METH + 1.0 \qquad (11)$$

$$ER = -.036 * METH + .006 * CMPLX + .86 \qquad (12)$$

To apply the model to the unused, eighteenth point, the base-line equation is first used to establish the standard effort. Since the estimated size of the project was 101,000 lines, this standard effort value was 163 man-months with a range for one standard error of from 130 to 204 man-months. When the additional attributes are used to compute the error ratio as given by the multiple regression equations, the results (for each of the above equations) are:

$$ER = -0.224$$

$$ER = -0.166$$

Converting these numbers back to multiplicative factors means dividing the standard effort by 1.224 and by 1.166, respectively. When these ratios are applied to the standard effort value, the revised effort values are found to be 133 man-months with a range for one standard error from 115 to 154 man-months for the first equation, and 140 man-months with a range for one standard error of from 121 to 162 man-months for the second equation. The actual effort for the project is known to have been 138 man-months.

Once any new project is added to the data base, at least the generation of the base-line relationship and the multiple regression analysis of the error ratios should be repeated. It may also be necessary to examine the factor groupings to see if they could be modified to increase the accuracy of the model or to include a previously unimportant attribute.

For our data, when this eighteenth point is added to the data base, the base-line equation becomes:

$$E = .73 * DL^{1.16} + 3.5 \quad (\text{s.e.e.}=1.25) \qquad (13)$$

while the equations to predict the error ratio from the attributes become:

$$ER = -.035 * METH + .98 \quad (s.e.e.=1.16) \quad (14)$$

$$ER = -.036 * METH + .009 * CMPLX + .80 \quad (s.e.e.=1.15) \quad (15)$$

It should be remembered that the original choice of factors from the entire set, and the groupings of these factors, was done with regard to predicting size as measured by developed lines and was not so specifically tuned to predicting developed modules. It is reasonable to expect, then, that the results of the models generated to predict effort from the number of developed modules using these attribute groupings will be less accurate than those using the number of developed lines. If the objective had been to generate a model specifically suited to predicting modules, various adjustments would have been made during the early part of the model's development. Also, it is advisable to review the model each time a new project is completed and its data is added to the data base. In this way the model can be refined and kept up-to-date, and will be able to take into account changes in the overall programming environment.

Although we are not reporting here the actual values and equations generated in the development of the other forms of this basic model (described under "Variations on the Model," above) it became apparent that none of the model types is by far better than the rest, especially considering the fact that they all have differing amounts of statistical significance. In terms of a purely investigative study, all of them should probably be examined further. As more environmental information is added to the data-base, it may be possible to reorganize the constituent groups involved in the environmental attributes and to produce better categories. Also, when several more projects are completed, it may be possible to justifiably expand the size of the set of variables used to predict the expected value in the multiple regression routine giving the potential for greater accuracy.

CONCLUSIONS

There is reason to believe that the techniques outlined here and used in our laboratory have potential in terms of producing a useful model which is specifically developed for use at any particular environment. The main difficulty seems to be in determining which environmental attributes really capture the reason for the differences in productivity among the projects. The use of too few of these attributes will mean less of the variation can possibly be explained, while the use of too many makes the analysis statistically meaningless. We found that it was necessary to stop including factors with the multiple regression analysis when the r-squared value indicated that we had explained no more than half of the variations among the error ratios. This would seem to indicate that there were considerably more influences upon the productivities of the projects than we managed to isolate. Simplifying the original idea for the model, however, which reduced the emphasis on the quality of the data did not weaken the accuracy of the model beyond useful proportions. This is particularly important when so much of the data which is essential to build the model is subjective and consequently non-linear.

Acknowledgements: The authors would like to thank Dr. Jerry Page of Computer Sciences Corporation and Frank McGarry of NASA/Goddard Space Flight Center for their invaluable help in providing the data for this study.

Research for this study was supported in part by National Aeronautics and Space Administration grant NSG-5123 to the University of Maryland. Computer time supported in part through the facilities of the Computer Science Center of the University of Maryland.

References

(1) Putnam, L., "A General Empirical Solution to the Macro Software Sizing and Estimating Problem," IEEE Transactions on Software Engineering 4, No. 4, 1978.

(2) Walston, C. and Felix, C., "A Method of Programming Measurement and Estimation," IBM Systems Journal 16, Number 1, 1977.

(3) Boehm, Barry W., Draft of book on Software Engineering Economics, to be published.

(4) Lawrence, M. J. and Jeffery, D. R., "Inter-organizational Comparison of Programming Productivity," Department of Information Systems, University of New South Wales, March 1979.

(5) Doty Associates, Inc., Software Cost Estimates Study, Vol. 1, RADC TR 77-220, June 1977.

(6) Wolverton, R., "The Cost of Developing Large Scale Software," IEEE Transactions on Computers 23, No. 6, 1974

(7) Aron, J., "Estimating Resources for Large Programming Systems," NATO Conference on Software Engineering Techniques, Mason Charter, N. Y. 1969.

(8) Carriere, W. M. and Thibodeau, R., "Development of a Logistics Software Cost Estimating Technique for Foreigh Military Sales," General Research Corporation, Santa Barbara, California, June 1979.

(9) Norden, Peter V., "Useful Tools for Project Management," Management of Production, M. K. Starr (Ed.) Penguin Books, Inc., Baltimore, Md. 1970, pp. 77-101.

(10) Basili, V. R. and Freburger, K, "Programming Measurement and Estimation in the Software Engineering Laboratory," Journal of Systems and Software, Vol. 2, No. 1, 1981.

(11) Basili, V. R. and Reiter, R. W. Jr., "An Investigation of Human Factors in Software Development," Computer Magazine, December 1979, pp. 21-38.

(12) Statistical Package for the Social Sciences, Univac 1100 series manual.

Appendix 1

Project	Effort (man-months)	Total Lines	New Lines	Developed Lines	Predicted Standard Effort	Effort Ratio Standard/ Actual	Method-ology	Complex-ity	Exper-ience
1	115.8	111.9	84.7	90.2	138.7	.835	30	21	16
2	96.0	55.2	44.0	46.2	65.8	1.459	20	21	14
3	79.0	50.9	45.3	46.5	66.2	1.194	19	21	16
4	90.8	75.4	49.3	54.5	79.0	1.150	20	29	16
5	39.6	75.4	20.1	31.1	42.9	.924	35	21	18
6	98.4	89.5	62.0	97.5	100.1	.982	29	29	14
7	18.9	14.9	12.2	12.8	17.5	1.082	26	25	16
8	10.3	14.3	9.6	10.5	14.7	.704	34	19	21
9	28.5	32.8	18.7	21.5	29.2	.977	31	27	20
10	7.0	5.5	2.5	3.1	6.2	1.128	26	18	6
11	9.0	4.5	4.2	4.2	7.4	1.220	19	23	12
12	7.3	9.7	7.4	7.8	11.4	.640	31	18	16
13	5.0	2.1	2.1	2.1	5.2	.957	28	19	20
14	8.4	5.2	4.9	5.0	8.2	1.025	29	21	14
15	98.7	85.4	76.9	78.6	118.8	.831	35	33	16
16	15.6	10.2	9.6	9.7	13.7	1.138	27	21	16
17	23.9	14.8	11.9	12.5	17.1	1.398	27	23	18
18	138.3	110.3	98.4	100.8	157.4	.879	34	33	16