12-12-1986

Study of the effectiveness of cost-estimation models and complexity metrics on small projects

Lein-Lein Chen
Florida International University

DOI: 10.25148/etd.FI14060166

ABSTRACT

STUDY OF THE EFFECTIVENESS OF COST ESTIMATION MODELS AND COMPLEXITY METRICS ON SMALL PROJECTS

by

Lein-Lein Chen

Software cost overruns and time delay are common occurrences in the software development process. To reduce the occurrences of these problems, software cost estimation models and software complexity metrics measurements are two popular approaches used by the industry.

Most of the related studies are conducted for large scale software projects. In this thesis, we have investigated the effectiveness of three popular cost estimation models and program complexity metrics in so far as their applicability to small scale projects is concerned.

Experiments conducted on the programs collected from FIU and NCR corporation indicate that none of the cost estimation models precisely estimates the actual development effort. However, the regression results indicate that the actual development effort is some function of the model variables. In addition, it also showed that the complexity metrics are useful measurements in predicting the actual development effort. Additional results related to lines of code metric are also discussed.

To Professor Jainendra Navlakha,

John Comfort,

Robert Fisher,

Samuel Shapiro,

This Thesis, having been approved in respect to form and mechanical execution, is referred to you for judgment upon its substantial merit.

Dean Professor James Mau
College of Arts and Sciences

The thesis of Lein-Lein Chen is approved.

Professor John Comfort

Professor Robert Fisher

Professor Samuel Shapiro

Major Professor Jainendra Navlakha

Date of Examination:

December 12, 1986

STUDY OF THE EFFECTIVENESS OF COST-ESTIMATION MODELS AND COMPLEXITY METRICS ON SMALL PROJECTS

by

Lein-Lein Chen

A thesis submitted in partial fulfillment of the

requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

at

FLORIDA INTERNATIONAL UNIVERSITY

November 1986

ACKNOWLEDGEMENTS

I wish to express my sincere gratitude to Dr. Jainendra Navlakha for his guidance throughout this work. In particular, his suggestions and reference aids were invaluable. I would also like to thank Dr. Carlos Brain, Dr. Paulette Johnson and especially Dr. Samuel Shapiro for sharing their knowledge of statistics and methods. My appreciation must also be extended to Dr. Wesley Mackey for familiarizing and authorizing my use of laser printer which made this printing possible.

In addition, my appreciation is also extended to all my friends and the faculty of Computer Science for their help, patience and data. And lastly, I am deeply indebted to Li Qiang for use of his metrics analyzer program which made part of the data collecting work much easier.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION AND BACKGROUND

Cost overruns and time delays in software development have been two major problems in the computer industry for a period of time. It is estimated that the total expenditures on all aspects of computing in the United State in 1980 was approximately 5 percent of the Gross National Product (GNP), or about $130 billion. It is further estimated that computing revenues will be 12.5 percent of the GNP by 1990 [7]. Thus the solution to the problem of cost overruns will provide tremendous financial benefits. In large-scale software development projects it is quite common for the costs to be double or triple the original estimate. Associated with these increasing software costs is the problem of time delays where up to 100 percent slippages have been quite common in the software development process [22]. This problem of late delivery can lead to a project's failure, and usually will increase the software development costs. On the other hand, as the software development costs have been increasing, the costs for hardware have been decreasing. In 1960, the ratio was approximately 80 percent hardware cost to 20 percent software cost for a system. By 1980, the ratio was reversed: approximately 20 percent hardware cost to 80 percent software cost, and by 1990, software costs will increase even more and will account for 90 percent of the amount spent on a computing system [7]. The rate of growth for the cost of software is greater than that of the United States economy in general [13]. The high

cost of software necessitates development of new techniques for quality software development and efficient cost control. During the past decade, many techniques have been proposed and/or developed in all areas of software engineering. One such development is the use of models and metrics based on the historical data and experience to quantitatively estimate the cost and provide better managerial control of projects. Two areas for research and development to address these issues have evolved. They are the development of cost estimation models which predict the costs of software, and the measurements of the attributes of software which enhance our understanding of the software development process.

A cost model is a formula or a set of formulas used to predict the costs likely to be incurred in a project [6]. Most of the cost models are similar in the sense that they are derived from the historical data base of an organization and use the number of lines of code (the number of source lines including or excluding comments) as the major factor to determine the cost of software. In addition, it is found that the environmental factors such as the ability of the personnel involved in developing the projects, techniques used (modern programming practices), target machines, languages, and application's complexity etc. also influence the software development cost. Some models incorporate such attibutes in the cost function as effort multipliers (also called cost drivers). Theoretically, there are many parameters that can affect the software costs. The general approach to cost modeling is to list all possible factors

and try to find a function by statistical analysis that performs well on the historical data. The model is built by selecting a set of variables which are significantly related to the variations of some specific attributes of the models such as productivity, effort etc. In addition to the cost of software which is determined by the amount of effort, other estimates such as software development time, manpower, risks, and trade-off etc. also can be computed by some of the cost estimation models. Cost models are very popular in industry, and they are being used by many organizations to predict the software development costs and schedules.

Software metrics are another important tool which help in one's understanding of the software development process. It is the area of software engineering that deals with techniques used to measure various properties of a software product. They can be used to measure many properties and attributes of software including its quality, complexity, productivity, reliability, maintainability, correctness, portability, and development effort. In addition, such measurements also can be used as a tool to manage resources and evaluate the quality of a design so that changes and improvements can be made during the software development process. Their importance in the software industry is quite evident. The general process to develop a software metric is as follows:

1. make some observations regarding an attribute of software;
2. hypothesize a set of principles to explain the

observations and intuitively define formulas for the metric;

3. perform experiments in a controlled environment to verify the accuracy of the metric; and
4. accept the metric as defined or make some adjustments to the hypotheses and formulas and loop back to step 3.

Clearly the development of software metrics is a trial and error process. Its accuracy depends on one's abiilty to perform good experiments in a controlled environment. Among all metrics, complexity metrics which measure the complexity of a program have gained maximum attention. Software complexity is the measure of the resources another system will expend while interacting with the software. It is also defined as the difficulty of manipulating software. One approach to define complexity metrics is based on the structure (or style) of the source code itself. In our research we concentrate on this type of complexity metrics, called program complexity metrics.

Almost all cost models [4,10,12,21,23,26,28] developed to date have been used for predicting the cost of software development of large projects. Very few experiments have been performed to test the applicability of these models for small projects. However there have been some studies [9] that have successfully applied complexity metrics to small projects. Since complexity metrics may measure the difficulty of the program, and since the actual development effort is proportional to the difficulty, hence the development effort should be related to these metrics.

Consequently, variations of the software costs from project to project should be explainable by the difference in program complexity as measured by the metrics. Present measurements of estimating the cost of software product are inadequate. Therefore it is desirable to search for new relationships which can be applied to the factors and which will result in more accurate predictions and hence better decisions may be made for controlling the cost and quality of the software system. This will improve the probability of success. We therefore, decided to conduct a study focusing on the following issues pertaining to the development of small projects:

a. The effectiveness of software cost estimation models- IBM, SLIM and COCOMO, and the relationship between the actual development effort and the predicted development effort computed by these models.
b. The relationship between complexity metrics and the actual development effort.
c. The relationship among the complexity metrics themselves.

This report is organized as follows. In section 2, three popular cost estimation models- IBM, SLIM and COCOMO which are relevant to our study are introduced. Section 3 reviews the software complexity metrics; two techniques- Halstead's software science equations and McCabe's cyclomatic complexity metrics are described. Section 4 proposes four research hypotheses and anticipated results. The details of the experiments performed investigating each hypothesis, the source of the data, descriptions of the

statistical analyses, the variables used in statistical analyses ,the data collection methodologies and tabular data appear in section 5. Next, the experimental results are discussed for each research hypothesis. Finally, section 7 contains the conclusions and some further inferences.

2. DESCRIPTION OF COST ESTIMATION MODELS

There exist many cost models in use in industry today such as SDC, DOTY, RCA PRICE, IBM Walston-Felix, Putnam's SLIM, Gruman's SOFCOST, Boehm's COCOMO, and Jensen's SEER, etc. [4,10,12,13,22,23,26,28,]. This study will be limited to three of these. They were selected because the details are available freely in the literature (many are classified), they are popular in the industry and are respresentatives of most of the existing models. The three models are IBM Walston-Felix, Putnam's SLIM, and Boehm's COCOMO.

2.1 IBM Walston-Felix Model 1977 [26]

The basic relationship between the number of lines of code and development effort was derived by Walston and Felix and is based on the study of sixty projects at IBM. The least squares fit to this data yields the result

$$E = 5.2 * (L)^{0.91}$$

where, E is the total effort in man months, and L is the size in thousands of lines of delivered source code including comments. In addition the authors also developed a measure of productivity. From the data base, they found twenty-nine variables (see Table 1) which showed significantly high correlations with productivity in their environment and were used in estimating the productivity index. The equation is defined as

$$I = \sum_{i=1}^{n} W_i * (X_i)$$

where

I: productivity index for a project

W_i: a factor weight based upon the productivity change for factor i

X_i: equals +1, 0, or -1 depending on whether the factor indicates increase, nominal, or decreased productivity

Other formulas given in this model include:

a) The relationship between documentation and delivered source lines is defined as

$$D = 49 * (L)^{1.01}$$

where, D is the number of pages of documentation, and L is source code in thousands of lines.

b) The relationship between project duration and delivered source lines is given by

$$M = 4.1 * (L)^{0.36}$$

where, M is the duration in months, and L is source code in thousands of lines.

c) The relationship between project duration and effort is as follows:

$$M = 2.47 * (E)^{0.35}$$

where, M is the duration in months, and E is total effort in man-months.

d) The relationship between average staff size and effort is as follows:

$$S = 0.54 * (E)^{0.6}$$

where, S is the average number of people on staff, and E is the total effort in man-months.

e) The relationship between computer cost and delivered source lines is given by

$$C = 1.84 * (L)^{0.96}$$

where, C is the computer cost in thousands of dollars and L is soruce code in thousands of lines. The constants in the above equations are derived from the historical data of IBM.

2.2 Putnam's SLIM Model 1978 [19-22]

This model is based on the empirical evidence that the pattern of life-cycle effort (in terms of man-year or man-month or man-hour) required to complete a large scale software project follows a Rayleigh distribution [25] (see Figure 1). Since this pattern was first shown by Norden (see Figure 2), it is also called the Rayleigh-Norden distribution [19]. Putnam empirically found that many medium to large scale software projects from different application areas exhibited the same life cycle pattern- a rise in manpower, a peaking and a tailing off (see Figure 3). For large systems, development cost is about 0.4 of the whole life cycle cost. The over all life-cycle curve of software development (see Figure 4) can be divided into stages such as systems definition, functional specification, software design, etc. Each stage provides better estimation than the previous one because more precise information about the size of the software system is available. The fundamental relationship among the source statements, the effort, the development time, and the state of the technology being applied to the project can be illustrated by the software equation

$$S = (C_k) * (K)^{1/3} * (T_d)^{4/3}$$

where

K: the total life-cycle effort in man-years;

T_d: the development time in years (time of peak manpower);

C_k: the state of technology constant which is calibrated from the organization's historical data. Its range for simple software cost estimating system is presented in Table 2.

S: the number of end product's delivered source lines of code.

Putnam used the technology constant to describe the environment under which the software was developed. This constant quantifies such factors as

.complexity of the system to be developed,

.development machine throughput capacity,

.software engineering tools,

.user interface (batch or interactive),

.target machine,

.development computer availability,

.discipline(modern programming practices),

.language,and

.human skills.

The total life cycle effort is given as

$$K = ((S / C_k)^3) / (T_d)^4$$

and the development effort is 40 percent of the total life-cycle effort. i.e.

$$E = 0.4 * (K)$$

That means, if everything is the same, then a large value of

technology constant will imply less development effort. From empirical data, Putnam also found that $K / (T_d)^2$ appeared to correspond to the difficulty of the system. Those systems which had been regarded as easy had small values of $K / (T_d)^2$, and those which had been regarded as hard had large values. He also found that the trade-off of effort and time can be explained by the software equation

$$K = \text{Constant} / (T_d)^4$$

From the relationship one sees that small changes in development time result in very large changes in effort. Putnam also mentioned that the development schedule can not be arbitrarily compressed by adding more resources in the system. The PERT sizing method is applied to estimate the size (in terms of number of source lines of code) of the software product at the beginning of the software development phase, which is then used with other parameters to determine the development effort as described before. A Monte Carlo simulation method is used to generate the milestones of the project in terms of fraction of total development time. Empirically studies give the following results:

EVENT	(t/T_d) MILESTONE FRACTION
Critical Design Review	.43
Systems Integration Test	.67
Prototype Test	.80
Start Installation	.93
Full Operation Capability	1.00

The same simulation method is also used to determine the risk (expressed in terms of probability) that a software product can be done within a specific value of cost, time and effort. For example, the probability of developing a specific project with five million dollars was 90% which is substantially higher than that with four million dollars (50%) [22].

Generally speaking, there exist many constraints such as contract delivery time, maximum peak man power available etc. that are imposed on the software product development. In the SLIM model, these constraint inequalities are expressed as functions of K and T_d. As all the equations are exponential in nature, they are linearized by taking logarithms. Linear programming techniques are used to determine the feasibility region from where management can perform cost/time tradeoffs for any project.

2.3 Boehm's Constructive Cost model (COCOMO) 1981 [2,3,4]

COCOMO was developed at TRW by Boehm. It represents a hierarchy of three models, Basic, Intermediate, and Detailed, which increase in precision. The model uses the number of team members, the project type and some other project and development environment characteristics as the basic variables. Each model may be applied in three different modes-Organic, Semidetached, and Embedded. The COCOMO estimating equations were obtained by analyzing a sample of sixty-three software projects in the data base. The Basic model uses only the number of instructions to

predict the development effort and hence its accuracy is only good enough for use in the early stages of a project. The nominal effort equations used in the Basic COCOMO are listed below:

Organic mode: $Effort(MM) = 2.4 * ((KDSI)^{1.05})$

Semidetached mode: $Effort(MM) = 3.0 * ((KDSI)^{1.12})$

Embedded mode: $Effort(MM) = 3.6 * ((KDSI)^{1.20})$

Where MM: man-months,

KDSI: source instruction in thousands of lines.

The Intermediate COCOMO model considers a set of fifteen variables which are called cost drivers or effort multipliers to explain much of the variations in software costs for different projects. These additional factors are grouped into the following four categories:

.Product attributes

RELY Required Software Reliability

DATA Data Base Size

CPLX Product Complexity

.Computer Attributes

TIME Execution Time Constraint

STOR Main Storage Constraint

VIRT Virtual Machine Volatility

TURN Computer Turnaround Time

.Personnel Attributes

ACAP Analyst Capability

AEXP Applications Experience

PCAP Programmer Capability

VEXP Virtual Machine Experience

LEXP Programming Language Experience

.Project Attributes

MODP Modern Programming Practices

TOOL Use of Software Tools

SCED Required Development Schedule

Each of these cost drivers (effort multipliers) is defined by a set of weights which depend on the particular project (see Table 3 and Table 4).

The general concepts of the Detailed COCOMO model are similar to the Intermediate COCOMO except that it decomposes the software product into module, subsystem, and system levels and uses phase sensitive effort multipliers for each cost driver attribute. These four phases are: Product Design (PD), Detail Design (DD), Code & Unit Test (CUT) and Integration & Test (IT). According to Boehm, some factors affect some phases much more than others. For example, projects with very high reliability requirements or hardware constraints will require more of an effort to integrate and test. Hence, for each of the effort multipliers, there is a weight corresponding to each phase, so the effort can be calculated by phase. The three level hierarchical decomposition of a software product from bottom to top is described below.

.Module level (lowest level)

It is described by the number of source instructions in the module, and by those cost drivers which tend to vary at the lowest level such as the module's complexity, the module programmers' capability and experience with the

language being used.

.Subsystem level

It is described by the remainder of the cost drivers such as time constraint, analyst capability, tools, schedule etc. which tend to vary from subsystem to subsystem.

.System level (top level)

It is a collection of all the subsystems.

The nominal effort equations of all modes listed below are used in both Intermediate and Detailed COCOMO models.

Organic mode: $Effort(MM) = 3.2 * ((KDSI)^{1.05})$

Semidetached mode: $Effort(MM) = 3.0 * ((KDSI)^{1.12})$

Embedded mode: $Effort(MM) = 2.8 * ((KDSI)^{1.20})$

where MM: man-months,

KDSI: source instruction in thousands of lines.

The main contribution that Detailed COCOMO provides beyond the Basic and Intermediate versions is a better basis for detailed project personnel planning with respect to the level of staff required to complete each development phase. In addition to estimating software cost from scratch, COCOMO also has formulas for caculating the effort for code adopted from the existing software. Other estimates such as development schedule and maintenance costs are also given.

All the above cost models were developed for estimating the cost of large size projects because it is the costliest component in the system. We now investigate their usefulness for small size projects.

3. REVIEW OF SOFTWARE COMPLEXITY METRICS

In the period prior to the mid 1970's, the only software complexity metric in use was the number of source lines of code. A system with more source lines was assumed to be more complex than another with less source lines. Starting in the mid 1970's, many other complexity metrics were developed, some were based on detailed and particular constructs of source code while the others were based on design structure chart and information flow in a system. The latter types by Yin & Winchester [29] and Henry & Kafura [11] are system complexty metrics and will not be considered here. This study will be limited to the metrics that depend on the source code or some particular features of the source code. Halstead [9] considered that each symbol of the code contributed towards software complexity, McCabe [14] assumed that only control flow transfers contributed to complexity while Albrecht [1] hypothesized that only I/O behavior and requirements determined the complexity of a software system. We describe the software complexity metrics of Halstead and McCabe which are relevant to our work.

3.1 Halstead's Software Science Equations [9]

Halstead considered each symbol of a program to be either an operator or an operand. The software science equations are based on the fundamental components of any program that are given below:

n_1: the number of distinct operators appearing in a program

n_2: the number of distinct operands appearing in a program

N_1: the total number of occurrences of the operators in a program

N_2: the total number of occurrences of the operands in a program

Based on these components, many other complexity metrics parameters are defined as follows:

Vocabulary $n = n_1 + n_2$

Length $N = N_1 + N_2$

Volume $V = N * \log_2 n$

where $\log_2 n$ is the number of bits required to store each symbol of the program and hence volume gives the total number of bits needed to store the whole program. The volume will change depending upon the power of the programming language, so it is possible for different implementations of an algorithm to have different volumes. The minimum volume is called Potential Volume (V^*) which is the property of the algorithm and is defined as

$$V^* = (n_1^* + n_2^*) * \log_2 (n_1^* + n_2^*)$$

where

n_1^*: represents the minimum number of distinct operators, ($n_1^* = 2$);

n_2^*: represents the minimum number of distinct operands to implement the algorithm. This number is very difficult to estimate.

Now, for a particular implementation of an algorithm, its level is determined by the ratio of potential volume of that

algorithm and the actual volume of the program, i.e.

$$L = V^* / V; \quad 0 < L <= 1.$$

The difficulty of implementation is given by

$$D = 1 / L$$

i.e. higher the level, less is the difficulty. As V^* is difficult to obtain (because n_2^* can not be determined), the estimated level based on the use of operands and operators is calculated as

$$L = (2 / n_1) * (n_2 / N_2).$$

The effort required to write the program (in terms of elementary mental discriminations or total number of "moments") is given by

$$E = V / L, \quad \text{or} \quad E = V * D.$$

The programming time T (in seconds) is defined as

$$T = E / S$$

where S is the Stroud factor and has units of "moments" per second. It was found by Stroud that a human brain is capable of performing between 5 to 20 elementary mental discriminations per second. Halstead used S = 18 mental discriminations per second in his studies.

3.2 McCabe's Cyclomatic Complexity Measure [14]

McCabe's cyclomatic complexity measure depends on the control flow of that program, that is, its decision structure. His overall strategy is to compute the number of linearly independent paths in the directed graph obtained from the program flow graph. However, a program with a backward branch could have infinite number of paths, so his

complexity measure is only defined in terms of the number of basic paths of a program. According to him the complexity measure, V(G), is correlated closely with the amount of work required to test a program based on the number of basic paths in it. He applied the properties of graph theory to the characteristics of the program structure, where the program itself corresponds to a directed graph with a single entry and single exit. The blocks of sequential code in the program then correspond to each node in the graph, and the branches which change the program control flows correspond to the arcs in the graph. The cyclomatic complexity of a strongly connected graph G is defined as

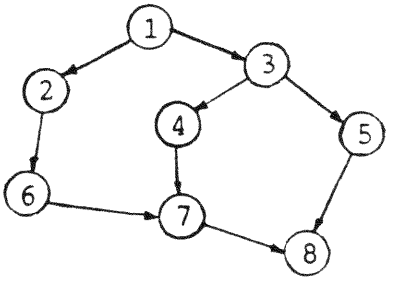
V(G) = Edges - Nodes + 2 * Components

where the terms Edges and Nodes represent the total number of edges and nodes in the graph respectively, and the term Components is the total number of connected components in the graph. The following two examples given by McCabe illustrate this idea.

Example 1:

V(G)= E - N + 2 * C
= 3 - 3 + 2 * 1
= 2

Example 2:



V(G)= E - N + 2 * C
= 9 - 8 + 2 * 1
= 3

It is also proved that in general the complexity of any program can be computed in terms of the number of simple predicates (decisions with single entry and single exit) in a program. Thus

$$V(G) = \# + 1$$

where # is the number of simple predicates in a program. Many variations to McCabe's complexity metrics exist. They all mainly differ from McCabe's metrics in the way in which conditions are counted. There is no evidence in the literature that they are better than McCabe's metrics and hence are not considered in this study. McCabe also observed that V(G) is a reasonable measure to compare and rank the complexity of various routines. An upper bound of 10 is recommened as the maximum complexity for the control graph of an individual routine.

4. RESEARCH HYPOTHESES

As mentioned earlier, this work focuses on the following issues regarding small scale software projects.

a. The effectiveness of software cost estimation models-IBM, SLIM and COCOMO, and the relationship between the actual development effort and the predicted development effort computed by these models.

b. The relationship between complexity metrics and actual development effort.

c. The relationship among the complexity metrics themselves.

To test the relationships among cost model parameters, software complexity metrics and software development effort, several hypotheses are made. These are as follows:

H1. The formulas given by IBM, SLIM and COCOMO models can be used to predict the actual development effort for small projects.

Cost models like IBM, SLIM, and COCOMO are popular models used in the industry for estimating the cost of large software projects. They have been shown to be very useful in resource management and cost control. However, no study has demonstrated that these models are also applicable to small projects. Since the basic formulas defined by these three models are all in terms of the program size, and small projects intuitively need less development effort, therefore we hypothesize that these models should also be usable for prediction of actual development effort for small scale software projects.

H2. McCabe's cyclomatic complexity is proportional to the

amount of actual development effort.

According to McCabe, the cyclomatic complexity of a program can be used to measure the difficulty of understanding and testing that program. Therefore, a program with more control paths than another will need more development effort for design, coding, debugging, and testing.

H3. Halstead's potential volume for an algorithm is proportional to the amount of actual effort required to implement it.

Halstead's potential volume measures the complexity of an algorithm in the sense that it is the minimum volume for an algorithm to be expressed in the most powerful language for that application. If potential volume increases, then the minimum volume of implementation of that algorithm increases which in turn means that the actual effort to implement it should also increase.

H4. The number of instructions is proportional to the total number of distinct symbols, the program volume, and the cyclomatic complexity of a program.

The number of source instructions has been commonly used as a measure of program size, and many predictions such as development effort and productivity are defined as a function of this measure. Since it is a very significant factor in all cost models and it is easy to compute, we investigated the relation of this attribute with other fundamental software complexity metrics.

Various descriptive measures from the source program

were obtained in order to test these hypotheses. The notations used for these measures are described below:

MEASURE	DESCRIPTION
DSI	The number of source lines of instruction excluding comments (called delivered source instructions).
SLOC	The number of source lines of code including comments.
n	Halstead's vocabulary size which is equal to the total number of distinct operators plus toal number of distinct operands.
N	Halstead's program length which represents the total occurrences of operators and operands appearing in a program.
V	Halstead's program volume.
V^*	Halstead's potential volume or minimum volume of an algorithm.
E	Halstead's program effort which represents the total number of elementary mental discriminations required for generating a program.
V(G)	McCabe's cyclomatic complexity of a program which may be expressed as the total number of predicates (conditions) plus one for a program.
DSI/V(G)	Which is the inverse of V(G) density, we call it instruction density.
ACTEFF	The actual effort to develop a software product.
IBMEFF	The development effort computed by IBM cost estimation model.

SLIMEFF The development effort computed by Putnam's SLIM cost estimation model.

COCOMOEFF The development effort computed by the Organic mode of Boehm's Intermediate COCOMO cost estimation model.

C_k Technology constant of the SLIM cost estimation model.

5. DETAILS OF EXPERIMENTS PERFORMED

In order to statistically test the hypotheses described in the previous section, experimental data was collected from two different environments: Florida International University and the National Cash Register Corporation (denoted FIU and NCR hereafter). The FIU data, which was collected in Spring 1986, consists of 54 PASCAL programs which were developed by students and professors in the Computer Science program. The NCR data, which was collected in August 1985, was from 324 modules of a large project written in NCRL at the NCR corporation. These modules were included because each is a small program by itself and the group provides a good comparison between programs written in a university and an industrial environment. Two data sets were produced from the FIU group. One set contains PASCAL programs with varying complexities ranging from the introductary level programs to graduate level compiler projects (see Table 5) and some other projects of individual professors and students. Several projects were selected from the same course and thus these all had the same objective. The second group consisted of a subset of the first. This group consisted of 28 distinct programs for which the corresponding development effort was recorded. In some instances the programmer specified the actual development effort while for class projects the professors approximated the actual development effort. The detailed description of the analyses performed and data collection process is described below.

5.1 Data Analysis

Regression analysis was used in this study for evaluating the previous stated hypotheses. In a regression analysis, the dependent (response) variable is expressed as a function of one or more independent (predictor) variables. The regression function estimates the average response for a given set of values of its independent variables. Thus it can be utilized to predict the average response for a given project.

Two types of regression models were used: simple linear regression and nonlinear regression. Computations were done using the software package BMDP (biomedical computer programs P-series) [5]. Another package SAS [24] was also used for plotting regression lines with 95% confidence intervals (see Figure 6-12). The adequacy of the simple linear regression model was measured by computing R^2 using the PRESS [16] technique.

5.1.1 Simple Linear Regression Model

The simple linear regression model uses a single regressor X, and is defined by

$$Y = B_0 + B_1X + e$$

where the intercept B_0 and the slope B_1 are unknown constants and e is a random error component. Since the parameters B_0 and B_1 are unknown and must be estimated using sample data, the method of Least Squares is used. This technique finds those estimates of B_0 and B_1 which minimize the residual sum of squares (sum of squared

differences between the observed and estimated values of Y). The linear dependency between the Y and X variables is tested by use of an F-statistic which is the ratio of the regression mean square, MSR, and residual mean square, MSE. The coeffient of determination, R^2, is defined to be the square of the ordinary correlation coefficient, r. It is used to indicate the proportion of the total variability in the response variable Y that can be accounted for by the predictor variable X [18].

5.1.2 Nonlinear Regression Model

A Nonlinear regression model is used when the relationship of the response to the independent variables can not be written in an expression which is linear in the parameters. The most general approach to the solution of a nonlinear problem is to attempt to get close enough to the best estimate, so that the nonlinear function may be approximated by the linear term in a Taylor series expansion. If this region can be entered, the problem has been linearized and the method of Least Squares may be used [15]. The Least Squares method estimates the parameters of a nonlinear function. For example, suppose the model is

$$Y = A(B)^X + e.$$

The model finds estimates of A and B by obtaining the values which minimize the sum of

$$(Y - A(B)^X)^2$$

taken over the sample values. Such method requires insertion of initial estimates of A and B. Many methods

have been developed in this field. One procedure which can be used with the above model is to obtain the initial estimates by using a log transformation to obtain a linear model and then using the least squares technique on the transformed model. This procedure was used in this analysis. The regression model produces the parameters of the nonlinear function and its residual mean square.

5.1.3. BMDP Statistics Package

This package provides regression analysis and data ploting programs. For linear regression, program P9R (all possible subsets regression) was used. This program was used because of the extensive residual analysis it provides, such as the PRESS statistics and its data plotting routines. The nonlinear regression analysis was done using the program P3R. This program estimates the parameters of a nonlinear function using Least Squares with a Gauss-Newton algorithm. The general formula which is selected by this program is defined below:

$$Y = P1\ (X)^{P2}\ (e)^{P3(X)} + \ldots$$

where

Y: dependent variable

X: independent variable

P1,P2,P3: constants

In this application, the user has to specify the initial values of the constants P1 and P2; P3 and e are ignored by the system. The initial estimates of P1 and P2 were obtained by first using a log transformation as described

above. Then the estimates of the intercept and slope were used as the initial values of P1 and P2 of the nonlinear function. The program P3R reached its smallest residual mean square with a small number of iterations.

5.1.4 PRESS Statistics

PRESS is one of the techniques used for evaluating regression models. The procedure works as follows: Select observation Y_i. Fit the regression model to the remaining n-1 observations and use the resulting equation to predict the withheld observation Y_i, denoting this predicted value $\tilde{Y}_{(i)}$. Find the prediction error for point i as

$$e_{(i)} = Y_i - \tilde{Y}_{(i)}.$$

The prediction error is often called the ith deleted residual. This procedure is repeated for each observation i=1,2,3..,n, producing a set of n deleted residuals $e_{(1)}, e_{(2)}, \ldots, e_{(n)}$. The PRESS statistic is defined as the sum of squares of the n deleted residuals,

$$\text{PRESS} = \sum_{i=1}^{n} e^2_{(i)} = \sum_{i=1}^{n} (Y_i - \tilde{Y}_{(i)})^2$$

The approximate R^2 for prediction is then defined as

$$R^2_{\text{pred}} = 1 - (\text{PRESS} / S_{YY})$$

where S_{YY} is called the corrected sum of squares of the

dependent variable. The BMDP statistic package, P9R, provides the information needed to calculate R^2_{pred}; the average deleted residual and the standard deviation of dependent variable, S_Y. By multiplying average deleted residual by n and S^2_Y by n-1, the values for PRESS and S_{YY} are obtained and the value of R^2_{pred} can then be obtained.

5.1.5 Proposed Statistical Analyses.

The following techniques were used to evaluate the various hypotheses. The observations were grouped into four different sets and analyzed as described below.

1. Analysis of the predicted development efforts produced by IBM,SLIM and COCOMO cost estimation models on small projects.

Two types of analyses were performed. The first compared the average relative error for each model. The average relative error was obtained by dividing the total relative errors by the number of projects. The relative error for each project was defined as follows:

$$\text{Relative Error} = \frac{\text{ABS(Actual- Predicted)}}{\text{Actual}}$$

where

ABS: absolute value,

Actual: actual development effort,

Predicted: predicted development effort.

A second analysis compared the squared correlations, R^2, between the actual development effort and the Predicted development effort for each model. The model error was

estimated from the regression model using the residual sum of squares. Since all the model predictions are basically a function of the program size, and since the actual effort is proportional to the program size, the predictions should be close to the actual results.

2. Analysis of the relationship between actual development effort and McCabe's cyclomatic complexity.

A regression model was used to relate the actual development effort (dependent variable) to McCabe's cyclomatic complexity (independent varible). There should be a strong positive relationship since the more complex a program the more development effort is needed.

3. Analysis of the relationship between Halstead's potential volume and actual development effort.

A Regression analysis was used to investigate the relationship of the actual development effort (dependent variable) and the potential volume (independent variable). Since the actual development effort should be proportional to the complexity of the algorithms, there should be a strong positive relationship.

4. Analysis of the relationship of the number of instructions (DSI) with the total number of distinct symbols (n), the program volume (V), and the cyclomatic complexity (V(G)).

It is intuitive that as the program size increases, so will the total number of distinct symbols, program volume, and the cyclomatic complexity, and vice versa. A regression analysis was performed on each pair using the number of

instructions (DSI) as the dependent variable and n, V, and V(G) as the independent variables respectively. A high correlation is expected for each pair.

5.2 Data Description

We have already identified the data required to perform the analysis described. The data collection methodologies and the description of the data collected are given below.

5.2.1 Data Collection Methodology

Programs were collected from FIU and NCR corporation. FIU programs included those from individual developers and students in selected classes. The data required from an individual's program was collected by interviewing the developer. Data on the class projects were obtained by consulting the instructor who approximated the development effort required. Since the instructor's approximates was used as the actual development effort of a completed project, only those projects that received 'A' or 'A-' grades were selected for the analysis. The data needed from the NCR source was available in a data base; however, this data did not include the actual development effort for each individual NCR module. Therefore these modules were excluded from the analysis relating to the model predictions.

5.2.1.1 Methodology Used for Collecting Complexity Metrics Parameters

The complexity metrics parameters of the FIU projects

were measured using a 'Metrics Analyzer' program. This program was developed by Qiang and Navlakha [17] for analyzing the complexity of a standard PASCAL program and was modified for this project. This program takes a syntactically correct and executable PASCAL program as input and generates the values of Halstead's complexity metrics as well as the number of source lines as its output (see Figure 5). The output of the Metrics Analyzer program for a sample program (see Appendix A) that contains most features of PASCAL, is shown in Appendix B.

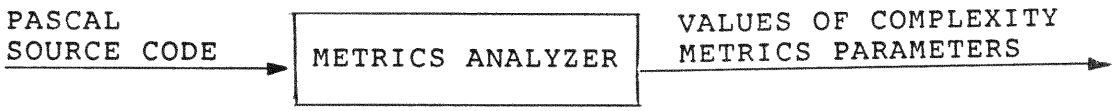


Figure 5. I/O Characteristics of the Metrics Analyzer Program

The counting and computation strategy for all the complexity metrics are straight forward. The one exception is McCabe's cyclomatic complexity, V(G). which is equal to the number of predicates plus one. In order to obtain the total number of predicates, the Metrics Analyzer program was modified to count the occurrences of each type of conditional statement such as IF, WHILE, FOR, CASE and REPEAT. For each of these, it determines the number of control flow paths (e.g. an IF statement increases the total control flow path count by one). The CASE statement is handled as follows.

1. Count the number of CASE clauses (the absence of 'ELSE' or 'OTHERWISE' increases the control flow path count by

one)

2. Substract one from it.

5.2.1.2 Methodology Used for Collecting the Information for the Cost Estimation Models

A cost estimation analysis was performed only for the FIU programs because the actual or estimated development efforts were known. In order to use the development time in the model, it was necessary to convert from school hours to man years or man months. We found that students average about ten hours a week on one project. Therefore 520 hours was used to represent a one-year working time, and this was used as a basis to convert hours to man-years or man-months for all the cost models.

The data collected could be used directly with the IBM cost model to calculate the development effort. However, for the SLIM and COCOMO models, values for the technology constant and effort multipliers were needed. In order to obtain the value for SLIM's C_k for the school environment, the constant 'C' is needed. This 'C' is used by Putnam in his simple software cost estimation system for selecting the C_k. The formula for 'C' is defined as

$$C = S_s / (E / B)^{1/3} (T_d)^{4/3}$$

where S_s is the number of executable source lines, E and T_d are the development effort in man-years and the development time in years respectively. 'B' is another constant derived by Putnam for computing 'C'. Its values

corresponding to a given project size are listed below.

SIZE(1000 LINES)	B
5 - 15	0.16
20	0.18
30	0.28
40	0.34
50	0.37
70	0.39
>= 100	0.39

To determine C_k, the following procedure is used:

.Convert 'E' and 'T_d' to units of years. In the school environment 520 hours was used to represent one-year working time for a project.

.Determine the value of B. There is no published value of B for projects of less than 5,000 lines required by this study and there is no published formula to calculate this quantity. Therefore the model development effort was calculated using values of 'B' ranging from 0.05 to 0.16. We observed that 0.16, which is the minimum value for 'B' corresponding to the source lines 5,000-10,000 produced good estimates for the development effort and hence was used in the calculation for 'C' for all projects.

.Compute 'C' from the function mentioned above.

.Select the value of C_k which is greater than and the closest to the computed 'C' from the values listed in Table 2. If no corresponding C_k exists, than 'C' is

used instead. The table has a range of C_k values applicable to a majority of industrial projects. Since the value of C is the basis for choosing C_k, this should be a valid technique.

The determination of the ratings for the cost driver attributes was done in consultation with the professors and programmers. In the COCOMO model, we basically retain Boehm's rating scale (see Tables 3 and Table 4) for most of the effort adjustment factors except some personnel attributes such as analysts' capability, application experience, programmer capability, virtual machine experience, and programming language experience. These constants were determined from considering the level of the course in the computer science program. For example, the introductary level PASCAL programs were rated "low" while the more sophiscated level operating system projects were rated "high" on the programming language experience attribute. Simple programs were written to do most of the needed calculations. The only exception was the calculation for the COCOMO models. An automated software package-WICOMO developed by the Wang Institute of Graduate Studies [27] was used. WICOMO is an interactive software cost estimation model available on IBM PC. It is based on Boehm's COCOMO model to calculate the estimates of man-month of effort, cost and schedule for a project. The major input parameters of WICOMO are project size and the ratings of environmental factors (i.e. effort multipliers).

5.2.2 Data Collected

Software complexity metrics parameter values (Halstead's parameters and McCabe's cyclomatic complexity) as well as total source lines and total instructions were obtained for all programs in both groups, FIU and NCR. However, only the FIU programs were used in the analysis related to the development effort.

Tables showing input from the FIU data group:

Table 6 lists the values of complexity metrics parameters for the 54 FIU PASCAL programs. Additionally, the instruction density (ratio of the number of instructions per V(G)), total number of source lines, the number of instructions and the source of program are also given. Calculations for McCabe's cyclomatic complexity, V(G), are shown in Table 7. The number of predicates is presented in the column labeled 'PREDICATE'. Complexity metrics values for programs obtained from the same class were averaged together. These average metrics values and the corresponding actual development effort are listed in Table 8, and marked with '*' in the first coulmn. Table 9 lists the parameters used to obtain the predicted development effort for the SLIM model. Table 10 includes the ratings for FIU projects and the predicted development effort obtained by using the Organic mode of Boehm's Intermediate COCOMO model. Table 11 gives the adjusted ratings for the personnel attributes used and gives the adjusted COCOMO model predicted effort.

Table showing input from the NCR data group:

The complexity metrics of NCR's 324 small programs are listed in Table 12. Some of the complexity metrics which were used for the effort prediction analysis (such as Halstead's potential volume, program effort and program time) are not needed for this analysis and hence are excluded from this table.

6. RESULTS

Table 13 contains the summary of actual and predicted development efforts which computed by all models for 28 FIU projects. For the predicted development efforts, Table 14 gives the relative errors in estimation ((actual - predicted) / actual) of all models for each project. It also includes the total, average and the range of the relative error of these predicted development efforts. Seven of these 28 projects are more than 1000 lines long and results for them are given in Table 15. The regression results using linear and nonlinear models on the FIU data are presented in Table 16 and Table 17, and those for NCR data are listed in Tables 18 and 19. These values indicated that the residual mean square computed from the linear regression model are close to those from the nonlinear model. This results from the fact that the regression relationship is close to linear, the power constant P2 is close to 1 (see Table 17 and Table 19). Various results regarding the previous hypotheses are discussed below:

6.1. The Effectiveness of Cost Estimation Models-IBM, SLIM, and COCOMO on Predicting Development Effort for Small Scale Projects

The accuracy of each model can be evaluated by comparing the actual and predicted development efforts. The analyses showed that the IBM and SLIM models usually overpredicted the actual effort whereas the COCOMO model

usually underpredicted the actual development effort (see Table 13). The average relative errors in estimation of these three models with respect to the actual development effort are 342%, 84%, and 56% (see Table 14). The range of the relative errors across all 28 FIU projects for IBM, SLIM and COCOMO models varied from 59% to 1559%, 18% to 150%, and 8% to 203% respectively. This indicated a large variability in the estimates in the case of small projects. The IBM model, which estimated the development effort by using the total number of source lines of code has the highest average relative error (342%) of all the models. The SLIM model, where development effort is a function of several parameters such as the number of source lines of instruction, technology constant, and development time had an average relative error of 84%; thus its accuracy is also poor. The Intermediate COCOMO model contains more variables than the IBM model and had smaller average relative error of 56% than either the SLIM or the IBM models; however it was still not satisfactory.

Since the accuracy of the COCOMO model depends on the proper choices of the values of the effort multipliers, and since many of its parameters do not account for the environment under which the small size projects were developed particularly in school, some adjustments were attempted to adapt it to this environment. The adjusted COCOMO model was constructed by considering only six of the fifteen effort multipliers which are used in the Organic mode of Boehm's Intermediate COCOMO model.

These six effort multipliers are

.applications experience,

.programmer capability,

.virtual machine experience,

.programming language experience,

.computer turnaround time,

.application type.

where, the last factor was adapted from RCA's PRICE model [8,23] with a slight change to suit this project. The modified application type and its ratings for the adjusted COCOMO are as follows:

APPLICATION TYPE	RATINGS
Operating Systems	Very High
Interactive Operations	Very High
String Manipulation	Nominal
Mathematical Operations	Low
Other	Very Low

As mentioned earlier, the given ratings were based on the course level in the computer science program. Operating systems and interactive operations projects were rated 'very high' because they both involve a considerable amount of design and programming time by students. Weights were assigned for each rating. These weights were developed by Boehm for one of his effort multipliers, Product complextity (CPLX) in the Intermediate COCOMO model. We did not use his CPLX definition because this factor is determined by a set

of sophisticated operations: program control, computation, device- dependent, and data management etc (see Table 3), which are not appropriate for a single module program in a school environment. The adjusted COCOMO model had an average relative error of 52% which is not much different than the Intermediate COCOMO model.

In addition to evaluating the cost estimation models, the same projects were also used to study the predicted development effort using Halstead's complexity metrics. This prediction was compared to the actual development effort, and the relative error in estimation varied between 8% to 83% with a mean relative error of 36%. This result is much better than that of all the previous models. Thus the complexity metrics appear to be a useful tool for measuring the development effort for small projects.

Further analysis of this data showed that projects with program lengths over 1000 lines have a smaller range of relative errors than the others. The prior analyses were repeated using only projects with lengths between 1100 and 2700 lines. The average relative errors of all models except COCOMO (which increased from 56% to 64%), were significantly reduced: IBM model error decreased from 342% to 195%, SLIM model error decreased from 84% to 58%,adjusted COCOMO model error decreased from 52% to 11%, and Halstead prediction error decreased from 36% to 19% (see Table 15). This indicates that the adjusted COCOMO model produced the smallest average relative error, 0.11 of all. It seems that Boehm's COCOMO model is more easily adaptable to a specific

organization and it is easier to tune this model to a particular environment. As the investigation shows, none of these cost models estimates development effort with a satisfactory precision for the small size projects in the school environment. It seems that the effort equations derived from a data base composed of large size projects are not useful for predicting the development effort for small projects directly.

In addition to studying the relative prediction error for small projects, the statistical significance of the regression analyses was examined. The regression results indicate that the predicted development efforts computed by all the above models are statistically significant with a p value of less than .001 (see Table 16). The plots of regression lines and the 95% confidence intervals about the regression line are shown in Figures 6 to 12. Some statistics are summarized in Table 20 below.

Table 20. Values of Correlation and Residual Mean Square between the Actual and Predicted Development Efforts of All Models on 28 FIU Projects

DEPENDENT VARIABLE	INDEPENDENT VARIABLE	LINEAR-MODEL R^2	LINEAR-MODEL MSE	NONLINEAR-MODEL R^2	NONLINEAR-MODEL MSE
ACTEFF	IBMEFF	0.8931	849.32	0.8913	863.66
ACTEFF	SLMEFF	0.8816	941.15	0.8811	944.72
ACTEFF	COCOMOEFF	0.7845	1712.22	0.7870	1691.76
ACTEFF	ADJUST-COCOMOEFF	0.9056	749.99	0.9090	723.28
ACTEFF	HLSTDEFF	0.9142	682.02	0.9085	726.82

The R^2 values listed in Table 20, except for the COCOMO model, are close to 90 percent using either the linear or nonlinear relationship. This means that approximately 90% of the variability in the actual development effort can be explained by the variables in the model. However the relative prediction error was quite high. In addition, the regression results also showed that the model residual mean square for the model development effort of adjusted COCOMO model and Halstead are smaller than the other models. The standard errors (square root of residual mean square) of COCOMO and Halstead models were 27.38 and 26.12 hours respectively. Thus, based upon the above regression results and the average relative errors indicated in Table 14, we conclude that both Halstead and adjusted COCOMO model give better predictions of the actual development effort than the other models.

6.2. The Relationship between Actual Development Effort and McCabe's Cyclomatic Complexity.

This relationship is described by the regression results listed on Table 16 and Table 17. The R^2 value for both the linear and non-linear models were approximately the same, 0.9003 and 0.9029 respectively. Thus the more the control paths the higher is the effort necessary for designing, coding, testing, and debugging a program, and therefore the higher is the required development effort. Thus the cyclomatic complexity is a useful measure of the required development effort.

6.3. The Relationship between Actual Development Effort and Halstead's Potential Volume.

The R^2 value for both the linear and nonlinear regression models were approximately the same, 0.7886 and 0.7871 respectively; both of which are significant. This shows that the development effort is related to the potential volume which is a measure of the difficulty of the algorithm. Although this R^2 was not as high as obtained from the prior models, it does indicate that the amount of effort required is a function of the complexity of the algorithm itself. Thus in the early design phase of software development when the algorithm is designed, it is possible to obtain an estimate of the development effort.

6.4. The Relationship between the Number of Instructions (DSI), the Total Number of Distinct Symbols (n), the Program Volume (V) and the Cyclomatic Complexity (V(G)).

These relationships can be examined from the results summarized in Table 21 below:

Table 21. Values of Correlation and Residual Mean Square between DSI and n, DSI and V, DSI and V(G) of FIU and NCR Programs

SOURCE	DENDENT VARIABLE	INDEPENDENT VARIABLE	LINEAR-MODEL R^2	LINEAR-MODEL MSE	NONLINEAR-MODEL R^2	NONLINEAR-MODEL MSE
F.I.U.	n	DSI	0.9291	3337.63	0.9175	3882.58
NCR	n	DSI	0.8166	8292.09	0.8604	6312.73
F.I.U.	V	DSI	0.9767	26825643.34	0.9877	14141900.00
NCR	V	DSI	0.9788	10332555.80	0.9792	10137100.00
F.I.U.	V(G)	DSI	0.9084	730.42	0.9052	757.75
NCR	V(G)	DSI	0.8760	448.72	0.8771	444.74

It shows that all the R^2 values are close to 0.90. While most of the R^2 values are close to each other there is a statistically significant difference ($p = 0.002$) between the two sources for the variables n and DSI. The R^2 for the FIU data of 0.9175 is higher than the NCR R^2 of 0.8604. Thus a higher percent of the variability in n was explained by DSI for the data that came from FIU. A Fisher's test of two coefficients of correlation was used for this evaluation. This could occur because all the FIU programs are composed of a single module. Every module is a complete entity which includes all the components of a project and performs a complete task. In contrast, some of the NCR programs contain only data declarations or trival function definitions which do not perform a complete task. Therefore, the relationship between DSI and n varied substantially from module to module. In general, these software metrics are highly correlated with the number of instructions. The results are in agreement with our early assumptions. In addition, the results indicated that Halstead's program volume varies substantially with the number of instructions. It seems that Halstead's program volume is a very useful attribute to measure the size of a small program. Since DSI is highly correlated with these complexity metrics, and since an estimate of the number of instructions (DSI) is available in the early phase of software development, the relationships between DSI and other complexity metrics (i.e. n, v, V(G)) which are derived from historical data enable the software developers

to obtain early information about the total effort required.

An interesting observation made from the experiment is that the mean instructions density (DSI/VG) for 54 FIU programs was 9.01 and for the 324 NCR programs was 11.53. The instruction densities for both the NCR and FIU data were studied by dividing them into groups according to the number of lines of instruction. Table 22 below shows that the mean instruction densities of both FIU and NCR projects decreased as the size of projects increased. Thus from these limited experiments it seems that the instruction density of a program with more than 1000 source lines tends to be constant and does not depend on the project type. If the ratio DSI/V(G) remains constant across different programming languages also, then this may reveal some inherent property of the languages.

Table 22. The Instruction Density(DSI/V(G)) of 54 FIU and 324 NCR Programs

PROJECT SIZE (DSI)	NBR OF PROJECTS	MEAN DSI	STD.DEV DSI	MEAN DSI/V(G)	STD.DEV DSI/V(G)
FIU 156 -2683(all)	54	637.80	554.47	9.01	2.76
FIU 100 - 500	28	293.11	92.10	10.17	2.85
FIU 501 -1000	17	626.53	98.36	8.08	2.16
FIU OVER 1000	9	1731.00	477.20	7.13	1.77
NCR 22 -3631(all)	324	361.15	452.38	11.53	8.52
NCR under100	97	57.45	24.05	12.11	7.11
NCR 100 - 500	155	239.10	113.20	11.97	10.45
NCR 501 -1000	40	684.48	137.48	10.46	4.69
NCR over 1000	32	1467.81	530.73	8.96	4.32

7. CONCLUSION

The experimental results indicate that our initial hypotheses were valid. We found that none of the formulas given by the IBM, SLIM, and COCOMO cost models accurately predicted the actual development effort for the 28 FIU projects. Apparently, environmental factors which were used by all the cost models in determing the development effort (e.g. SLIM's technology constant, COCOMO's effort multipliers) are not applicable for small software product development. This is evident from the results of the adjusted COCOMO model which excluded these factors and which resulted in a more accurate estimate with smaller relative error and standard error than the other cost estimation models. This was particularly true for projects larger than 1000 lines of instruction. Although the relative error for all of the models was high, the results of the regression always indicated that actual development effort is some function of the model variables. Thus the regression functions derived from using the models might be applicable for predicting software costs.

We also found for the 28 projects developed at FIU that using Halstead's effort to predict effort gave smallest standard error and the highest correlation with the actual development effort. We conclude that as compared with the large projects the development effort for small projects are more dependent on the source code itself. We also found a strong correlation between McCabe's cyclomatic complexity and the actual development effort. Hence, software

complexity metrics seem to be reasonable measurements which can be used to estimate the amount of effort required to develop and test the software product.

This study also demonstrated that the complexity of an algorithm is related to the actual development effort. Thus, in the early software design phase, once an algorithm is specified the development effort can be predicted. Our experiments indicated that the number of source lines of instruction is strongly correlated with Halstead's vocabulary, program volume, and McCabe's cyclomatic complexity respectively. There exist techniques to estimate lines of code at the beginning of a project and hence estimate the values of various complexity metrics. Thus the regression functions derived can be used to predict values of other software attributes. The final observation from the experiments is that the instruction density of programs with lengths of more than 1000 lines of instructions appears to be constant. This shows that the instruction density can be used to estimate the DSI once the detailed design is done and V(G) is available. If this estimation of DSI is accurate, then it facilitates the prediction of other software attributes related to DSI more accurately. However, this is not established conclusively by this research and is a topic for future research.

Table 1. IBM 29 Variables that Correlate Significantly with Programming Productivity

The programming productivity is defined as the ratio of the delivered source lines of code (DSL) to the total effort in man- months (MM). Following twenty-nine variables were combined into an index based on the effect of each variable on productivity, and the analysis was performed on each variable independently. It does not take into account the possibility that these variables may be correlated.

1. Customer interface complexity
   a. Normal (500 DSL/MM)
   b. Greater than normal (124 DSL/MM)
   c. Less than normal (500 DSL/MM)

2. User participation in the definition of requirements
   a. None (491 DSL/MM)
   b. Some (267 DSL/MM)
   c. Much (205 DSL/MM)

3. Customer originated program design changes
   a. Few (297 DSL/MM)
   b. Many (196 DSL/MM)

4. Customer experience with the application area of the project
   a. NOne (318 DSL/MM)
   b. SOme (340 DSL/MM)
   c. Much (206 DSL/MM)

5. Overall personnel experience and qualifications
   a. Low (132 DSL/MM)
   b. Average (257 DSL/MM)
   c. High (410 DSL/MM)

6. Percentage of programmers doing development who participated in design of functional specifications
   a. Less than 25% (153 DSL/MM)
   b. 25% - 50% (242 DSL/MM)
   c. Greater than 50% (391 DSL/MM)

7. Previous experience with operational computer
   a. Minimal (146 DSL/MM)
   b. Average (270 DSL/MM)
   c. Extensive (312 DSL/MM)

8. Previous experience with programming languages
   a. Minimal (122 DSL/MM)
   b. Average (225 DSL/MM)
   c. Extensive (385 DSL/MM)

9. Previous experience with application of similar or greater size and complexity
   a. Minimal (146 DSL/MM)
   b. Average (221 DSL/MM)
   c. Extensive (410 DSL/MM)

10.Ratio of average staff size to duration (people/month)
   a.Less than 0.5 (305 DSL/MM)
   b.0.5 - 0.9 (310 DSL/MM)
   c.Greater than 0.9 (173 DSL/MM)

11.Hardware under concurrent development
   a.No (297 DSL/MM)
   b.Yes (177 DSL/MM)

12.Development computer access, open under special request
   a.0% (226 DSL/MM)
   b.1 - 25% (274 DSL/MM)
   c.Greater than 25% (357 DSL/MM)

13.Development computer access, closed
   a.0 -10% (303 DSL/MM)
   b.11 - 85% (251 DSL/MM)
   c.Greater than 85% (170 DSL/MM)

14.Classified security environment for computer and 25% of programs and data
   a.No (289 DSL/MM)
   b.Yes (156 DSL/MM)

15.Structured programming
   a.0 - 33% (169 DSL/MM)
   b.34 - 66%
   c.66% (301 DSL/MM)

16.Design and code inspections
   a.0 - 33% (220 DSL/MM)
   b.34 - 66% (300 DSL/MM)
   c.Greater than 66% (339 DSL/MM)

17.Top down development
   a.0 -33% (196 DSL/MM)
   b.34 -66% (237 DSL/MM)
   c.Greater than 66% (321 DSL/MM)

18.Chief programmer team usage
   a.0 -33% (219 DSL/MM)
   b.34 - 66%
   c.Greater than 66% (408 DSL/MM)

19.Overall complexity of code developed
   a.Less than average (314 DSL/MM)
   b.Greater than average (185 DSL/MM)

20.Complexity of application processing
   a.Less than average (349 DSL/MM)
   b.Average (345 DSL/MM)
   c.Greater than average (168 DSL/MM)

21.Complexity of program flow
   a.Less than average (289 DSL/MM)
   b.Average (299 DSL/MM)
   c.Greater than average (209 DSL/MM)

22.Overall constraints on program design
   a.Minimal (293 DSL/MM)
   b.Average (286 DSL/MM)
   c.Severe (166 DSL/MM)

23.Program design constraints on main storage
   a.Minimal (293 DSL/MM)
   b.Average (277 DSL/MM)
   c.Severe (193 DSL/MM)

24.Program design constraints on timing
   a.Minimal (303 DSL/MM)
   b.Average (317 DSL/MM)
   c.Severe (171 DSL/MM)

25.COde for real time or interactive operation, or executing under severe time constraint
   a.Less than 10% (279 DSL/MM)
   b.10 - 40% (337 DSL/MM)
   c.Greater than 40% (203 DSL/MM)

26.Percentage of code for delivery
   a.0 - 90% (159 DSL/MM)
   b.91 - 99% (327 DSL/MM)
   c.100% (265 DSL/MM)

27.COde classified as non-mathematical application and I/O programs
   a.0 - 33% (188 DSL/MM)
   b.34 - 66% (311 DSL/MM)
   c.67 - 100% (267 DSL/MM)

28.Number of classes of items in the data base per 1000 lines of code
   a.0 - 15 (334 DSL/MM)
   b.16 - 80 (243 DSL/MM)
   c.Greater than 80 (193 DSL/MM)

29.Number of pages of delivered documentation per 1000 lines of delivered code
   a.0 - 32 (320 DSL/MM)
   b.33 - 88 (252 DSL/MM)
   c.Greater than 88 (195 DSL/MM)

Table 2. Range of Technology Constant of a Very Simple Software Cost Estimation System Developed by Putnam

NBR	RANGE OF C_k
1	610
2	754
3	987
4	1220
5	1597
6	1974
7	2584
8	3194
9	4181
10	5168
11	6765
12	8362
13	10946
14	13530
15	17711
16	21892
17	28657
18	35422
19	46368

where

C_k: technology constant

Table 3. Fifteen Effort Multipliers of Boehm's Intermediate COCOMO Model

Product attributes:

RELY(Required software reliability):

Rating	Description
Very Low:	slightly inconvenience
Low:	Low easily recoverable losses
Nominal:	Moderate recoverable losses
High:	High financial loss
Very High:	Risk to human life

DATA(Data base size):
The data base size refers to the amount of data to be assembled and stored in nonmain storage by the time of software acceptance.

Rating	Description
Low:	DB bytes / Prog dsi < 10
Nominal:	10 <= D / P < 100
High:	100 <= D / P < 1000
Very High:	D / P >= 1000

CPLX(Software product complexity):
The ratings are a function of the type of operations to be primarily performed by the module. It includes control, computation, device-dependent, and data management operations.

Control:

Rating	Description
Very Low:	Straight line code with a few non-nested operaors: DOs,CASEs, IF-THEN-ELSEs, simple pridicates.
Low:	Straight forward nesting of operators. Mostly simple predicates.
Nominal:	Mostly simple nesting. Some intermodule control and use of decision tables.
High:	Highly nest SP operators with many compound predicates, 'QUEUE' and 'STACK' control. Considerable intermodule control.
Very High:	Reentrant and recursive coding. Fixed priority interrupt handling.
Extra High:	Multiple resource scheduling with dynamically changing priorities. Microcode-level control.

Computational operations:

Rating	Description
Very Low:	Evaluation of simple expressions:

A = B + C * (D - E)

Low: Evaluation of moderate level expressions:
D = SORT (B **2 - 4 * A * C)

Nominal: Use of standard math and statistical routines. Basic matrix and vector operations.

High: Basic numerical analysis: multivariate interpolation, ordinary differential equations.

Very High: Difficult but structured numerical analysis, near-singular matrix equations, partial differential equations.

Extra High: Difficult and unstructured numerical analysis, highly accurate analysis of noisy, stochastic data.

Device-dependent operations:

Rating	Description
Very Low:	Simple read, write statements with simple formats.
Low:	No cognizance needed of particular processor or I/O device characteristics. I/O done at GET/PUT level. No cognizance of overlap.
Nominal:	I/O processing includes device selection, status checking and error processing.
High:	Operations at physical I/O level (physical storage address translations, seeks, reads).
Very High:	Routine for interrupt diagnosis, servicing, masking and communication line handling.
Extra High:	Device timing dependent coding, microprogrammed operations.

Data management operation:

Rating	Description
Very Low:	Simple arrays in main memory
Low:	Single file subsetting with no data structure change, no edit, no intermediate file.
Nominal:	Multifile input and single file output. Simple structural changes, simple edits.
High:	Special purpose subroutines activated by data stream contents. Complex data restructuring at record level.
Very High:	A generalized, parameter-driven file structuring routine. File building, command processing, search optimization.
Extra High:	Highly coupled, dynamic relational structures and natural language data management.

Computer attributes:

TIME(Execution time constraint):

Rating	Description
Nominal:	<= 50% use of available execution time
High:	70%
Very High:	85%
Extra High:	95%

STOR(Main storage constraints):

Rating	Description
Nominal:	<= 50% use of available storage
High:	70%
Very High	85%
Extra High:	95%

VIRT(Virtual machine volatility):

It refers to the level of volatility of the virtual machine underlying the subsystem to be developed. For a given software subsystem, the underlying virtual machine is the complex of hardware and software that the subsystem calls upon to accomplish its task.

Rating	Description
Low:	Major change every 12 months
	Minor: 1 month
Nominal:	Major: 6 months
	Minor: 2 weeks
High:	Major: 2 months
	Minor: 1 week
Very High:	Major: 2 weeks
	Minor: 2 days

TURN(Computer turnaround time):

Rating	Description
Low:	Interactive
Nominal:	Average turnaround < 4 Hours
High:	4-12 hours
Very High:	> 12 hours

Personnel attributes:

ACAP(Analyst capability):

Rating	Description
Very Low:	15th percentile
Low:	35th percentile
Nominal:	55th perecntile

High:	75th percentile
Very High:	90th percentile

AEXP(Applications experience):

Rating	Description
Very Low:	<= 4 months experience
Low:	1 Year
Nominal:	3 Years
High:	6 Years
Very High:	12 Years

PCAP(Programmer capability):

Rating	Description
Very Low:	15th percentile
Low:	35th percentile
Nominal:	55th percentile
High:	75th percentile
Very High:	90th percentile

VEXP(Virtual machine experience):

Rating	Description
Very Low:	<= 1 month experience
Low:	4 months
Nominal:	1 year
High:	3 years

LEXP(Programming language experience):

Rating	Description
Very Low:	<= 1 month experience
Low:	4 months
Norminal:	1 year
High:	3 years

Project attributes:

MODP(Use of modern programming practices):

Rating	Description
Very Low:	No use
Low:	Beginning use
Nominal:	Some use
High:	General use
Very High:	Routine use

TOOL(Use of software tool):

Rating	Description
Very Low:	Basic microprocessor tools
Low:	Basic minicomputer tools
Nominal:	Basic mini/maxi computer tools
High:	Strong maxicomputer programming, test tools
Very High:	Advanced maxicomputer tools.

SCED(Schedule constraint):

Rating	Description
Very Low:	75% (severe acceleration)
Low:	85% (moderate acceleration)
Nominal:	100% (according to schedule)
High:	130% (moderate stretchout)
Very High:	160% (severe stretchout)

Table 4. Ratings of COCOMO Development Effort Multipliers

ATTRIBUTES	WEIGHT					
	VERY LOW	LOW	NOMINAL	HIGH	VERY HIGH	EXTRA HIGH
RELY	.75	.88	1.00	1.15	1.40	-
DATA	-	.94	1.00	1.08	1.16	-
CPLX	.70	.85	1.00	1.15	1.30	1.65
TIME	-	-	1.00	1.11	1.30	1.66
STOR	-	-	1.00	1.06	1.21	1.56
VIRT	-	.87	1.00	1.15	1.30	-
TURN	-	.87	1.00	1.07	1.15	-
ACAP	1.46	1.19	1.00	.86	.71	-
AEXP	1.29	1.13	1.00	.91	.82	-
PCAP	1.42	1.17	1.00	.86	.70	-
VEXP	1.21	1.10	1.00	.90	-	-
LEXP	1.14	1.07	1.00	.95	-	-
MODP	1.24	1.10	1.00	.91	.82	-
TOOL	1.24	1.10	1.00	.91	.83	-
SCED	1.23	1.08	1.00	1.04	1.10	-

where

RELY:Required Software Reliability
DATA:Data Base Size
CPLX:Product Complexity
TIME:Execution Time Constraint
STOR:Main Storage Constraint
VIRT:Virtual Machine Volatility
TURN:Computer Turnaround Time
ACAP:Analyst Capability
AEXP:Aplications Experience
PCAP:Programmer Capability
VEXP:Virtual Machine Experience
LEXP:Programming Language Experience
MODP:Use Of Modern Programming Practices
TOOL:Use Of Software Tools
SCED:Required Development Schedule

TABLE 5. FIU Projects' Types

COP# 3180(Programming in PASCAL)
COP# 3530(Data Structures)
COP# 4101(Structured Computer Organization)
COP# 4610(Operating Systems Principles)
COP# 4637(Introduction to Software Engineering)
COP# 5640(Compiler Construction)
MAD# 5405(Numeric Methods)

Table 6. Complexity Metrics Parameters of FIU Projects - 54 PASCAL Programs

NBR	DSI	SLOC	n	N	V	V^*	E	T	V(G)	DSI/VG	CLASS
1	156	309	145	762	5471.09	102.66	291584.26	4.49	16	9.75	DS.KK
2	180	309	125	826	5753.74	91.33	362485.48	5.59	21	8.57	DS.LF
3	184	298	135	811	5739.30	98.03	336009.77	5.18	17	10.82	DS.KK
4	194	261	108	936	6322.57	110.49	361809.33	5.58	10	19.40	MACKY
5	200	368	148	1366	9631.83	116.34	797389.86	12.31	16	12.50	DS.KB
6	200	304	129	1377	9654.46	88.19	1056855.23	16.31	20	10.00	DS.KB
7	201	304	155	1064	7741.80	122.74	488318.68	7.54	17	11.82	DS.KK
8	201	519	144	1010	7241.62	140.39	373515.36	5.76	22	9.14	DS.LB
9	208	257	156	1484	10811.54	129.16	905004.74	16.31	29	7.17	DS.KB
10	218	294	161	1452	10644.49	140.69	805350.65	12.42	20	10.90	DS.KB
11	224	322	148	1016	7324.80	114.15	470008.30	7.25	29	7.72	LOOK
12	259	325	130	1571	11302.14	97.19	1252262.79	19.33	38	6.82	DS.KB
13	269	356	172	1490	11065.13	163.67	748073.73	11.54	30	8.97	DS.LG
14	291	362	179	1385	10365.08	148.33	724296.30	11.17	32	9.09	DS.LG
15	309	420	178	1462	10929.52	174.97	682720.83	10.53	20	15.45	OS.B
16	322	906	186	2291	17272.21	247.43	1205733.33	18.61	41	7.85	SFENG
17	326	665	217	1801	13978.55	224.79	869239.35	13.41	38	8.58	DS.LB
18	328	429	199	1898	14494.31	179.44	1170813.52	18.06	57	5.75	DS.LB
19	328	429	202	1703	13041.93	192.08	885531.80	13.67	30	10.93	DS.LG
20	342	838	291	1749	14315.34	288.07	711388.53	10.97	35	9.77	ORGZN
21	358	371	140	1640	11692.02	118.29	1155707.41	17.83	46	7.78	PASCL
22	387	451	162	1807	13263.11	159.86	1100389.29	16.98	38	10.18	OS.B
23	392	679	268	2113	17043.65	243.50	1192981.46	18.41	33	11.88	OS.LI
24	403	549	143	1888	13517.84	114.37	1597783.97	24.66	39	10.33	DS.KB
25	408	482	188	1761	13303.63	155.17	1140611.30	17.60	55	7.42	DS.LB
26	416	616	243	1893	15001.67	292.86	768450.03	11.86	31	13.42	OS.B
27	423	646	218	1889	14681.87	229.71	938376.62	14.48	32	13.22	OS.B
28	480	592	254	2476	19779.98	224.82	1740273.91	40.28	50	9.60	UPDTE
29	502	922	241	2279	18033.48	216.83	1499798.58	23.14	54	9.30	SFENG
30	512	655	244	2443	19374.79	236.17	1589478.07	24.52	53	9.66	PASCL
31	513	854	185	2295	17284.52	153.35	1948162.68	30.06	86	5.97	SFENG
32	535	668	330	2221	18581.60	330.25	1045499.50	16.13	71	7.54	PRABU
33	550	977	276	2367	19192.88	278.81	1321202.65	20.39	41	13.41	OS.M
34	554	945	203	2353	18504.43	253.64	1349986.50	20.83	51	10.86	SFENG
35	593	917	242	2922	23138.92	222.79	2403240.16	37.09	93	6.38	SPELL
36	620	697	268	3279	26448.71	204.80	3415611.57	52.71	116	5.34	MTRXF
37	621	688	198	3435	26206.84	147.92	4643196.75	71.65	121	5.17	MTRXF
38	642	845	293	4189	34327.83	326.71	3606921.75	55.67	81	7.93	ORGZN
39	656	720	277	3000	24341.23	276.66	2141623.88	33.04	88	7.45	PASCL
40	663	803	302	3481	28677.89	368.58	2231351.97	34.43	97	6.84	PASCL
41	674	1292	242	4479	35468.59	253.19	4968600.00	76.67	68	9.91	ORGZN
42	677	1083	269	3601	29065.34	243.48	3469642.88	53.54	81	8.36	OS.LW
43	720	841	289	4238	34645.33	274.78	4368176.57	67.41	88	8.18	PRABU
44	740	987	315	4162	34541.30	245.94	4851077.50	74.86	124	5.97	MARXL
45	879	1566	557	5598	51062.34	668.65	3899431.54	60.18	97	9.06	OS.HL
46	1139	1471	562	6260	57181.51	563.89	5798534.75	89.48	112	10.17	PRABU
47	1309	1934	387	6606	56786.43	369.10	8736649.00	134.82	209	6.26	MTRXF
48	1311	1579	436	6505	57037.04	415.70	7825830.00	120.77	231	5.68	REVEW

49	1653	2579	619	10099	93656.06	608.27	14420350.55	222.54	179	9.23	PASER
50	1716	2196	701	10564	99864.35	697.16	14305188.67	220.75	364	4.71	FSHER
51	1786	3734	738	10355	98657.02	740.63	13141825.43	202.81	253	7.06	NCRCN
52	1821	2332	626	10267	95380.62	628.70	14470253.18	223.31	268	6.79	CMP.L
53	2165	3742	965	12560	124524.68	1076.20	14408531.67	222.35	361	6.00	STEVE
54	2683	4068	1163	16655	169608.45	1283.73	22409023.30	345.82	325	8.26	CMP.C

---

where

DSI: the number of source line of instructions excluding comments

SLOC: the number of source lines including comments

n: Halstead's vocabulary (i.e. total number of distinct operators and operands)

N: Halstead's program length (i.e. total number of occurrences of operands and operators)

V: Halstead's program volume

V^*: Halstead's potential volume of an algorithm

E: Halstead's total number of elementary mental discriminations required to generate a program (i.e. program effort)

T: Halstead's estimated program effort in terms of hours

V(G): McCabe's cyclomatic complexity(i.e.basic program control paths) which is equal to the number of predicates plus one

DSI/VG:the number of sequential instruction per basic control path

CLASS: source of program

Table 7. Calculations of McCabe's Cyclomatic Complexity for FIU 54 Programs

NBR	DSI	IF	CASE	WHILE	FOR	REPEAT	PREDICATE	V(G)
1.	156	10	–	1	3	1	15	16
2.	180	14	–	1	4	1	20	21
3.	184	11	–	2	3	–	16	17
4.	194	4	3	2	–	–	9	10
5.	200	12	–	1	–	2	15	16
6.	200	5	–	2	10	2	19	20
7.	201	9	–	4	3	–	16	17
8.	201	15	–	6	–	–	21	22
9.	208	13	–	6	9	–	28	29
10.	218	6	–	3	10	–	19	20
11.	224	13	2	3	9	1	28	29
12.	259	22	–	2	13	–	37	38
13.	269	14	–	7	8	–	29	30
14.	291	20	–	1	6	4	31	32
15.	309	11	–	1	4	3	19	20
16.	322	30	–	3	7	–	40	41
17.	326	21	–	3	7	6	37	38
18.	328	46	–	4	1	5	56	57
19.	328	16	2	3	7	1	29	30
20.	342	8	12	–	12	2	34	35
21.	358	20	10	4	11	–	45	46
22.	387	31	–	3	3	–	37	38
23.	392	18	3	5	5	1	32	33
24.	403	16	–	22	–	–	38	39
25.	408	46	–	7	1	–	54	55
26.	416	23	–	3	4	–	30	31
27.	423	20	–	10	1	–	31	32
28.	480	38	–	7	4	–	49	50
29.	502	34	–	13	5	1	53	54
30.	512	28	8	8	7	1	52	53
31.	513	71	–	3	9	2	85	86
32.	535	28	23	1	18	–	70	71
33.	550	30	–	–	10	–	40	41
34.	554	35	–	9	6	0	50	51
35.	593	59	17	11	3	2	92	93
36.	620	55	15	8	29	8	115	116
37.	621	63	–	10	43	4	120	121
38.	642	57	5	–	17	1	80	81
39.	656	37	28	5	7	10	87	88
40.	663	30	29	3	33	1	96	97
41.	674	23	28	2	14	–	67	68
42.	676	56	8	9	7	–	80	81
43.	720	46	28	1	12	–	87	88
44.	740	61	11	12	33	6	123	124
45.	879	74	9	3	10	–	96	97
46.	1139	53	12	27	11	8	111	112
47.	1309	122	–	28	58	–	208	209

48.	1311	129	36	35	25	5	230	231
49.	1653	85	53	18	12	10	178	179
50.	1716	238	81	26	5	13	363	364
51.	1786	123	66	35	15	13	252	253
52.	1821	50	186	14	12	5	267	268
53.	2165	179	61	19	95	6	360	361
54	2683	142	130	36	6	10	324	325

---

where

DSI: the number of source line of instructions excluding comments

IF: the number of 'IF' statements in a program

CASE: the number of 'CASE'clauses in a program

WHILE: the number of 'WHILE' statements in a program

FOR: the number of 'FOR' statements in a program

REPEAT:the number of 'REPEAT' statement in a program

PREDICATE: total number of 'IF', 'CASE', 'WHILE','FOR', and 'REPEAT' in a program (i.e. total number of predicates)

V(G): McCabe's cyclomatic complexity (i.e. program control paths) which is equal to the number of predicates plus one

Table 8. Complexity Metrics and Actual Development Effort of 28 FIU Projects

	NBR	DSI	ACT	SLOC	n	N	V	V^*	E	T	V(G)	CLASS
*	1.	180	20	304	145	879	6317.40	107.81	371970.90	5.74	16	DS.KK
	2.	180	9	309	125	826	5753.74	91.33	362485.48	5.59	21	DS.LF
	3.	194	4	261	108	936	6322.57	110.49	361809.33	5.58	10	MACKY
	4.	224	30	322	148	1016	7324.80	114.15	470008.30	7.25	29	LOOK
*	5.	248	15	350	145	1518	9125.13	92.80	1069107.89	16.66	29	DS.KB
	6.	296	18	382	184	1526	11490.71	168.02	785967.27	12.13	31	DS.LG
*	7.	316	22	524	187	1618	12254.53	174.95	888544.88	13.71	43	DS.LB
*	8.	384	80	533	201	1763	13469.04	214.35	872484.19	13.46	30	OS.BK
	9.	392	40	679	268	2113	17043.65	243.50	1192981.46	18.41	33	OS.LI
*	10.	473	80	907	204	2304	17773.66	217.79	1500920.27	23.16	58	SFENG
	11.	480	50	592	254	2476	19779.98	224.82	1740273.91	40.28	50	UPDTE
	12.	535	30	668	330	2221	18581.60	330.25	1045499.50	16.13	71	PRABU
*	13.	547	20	637	240	2641	21021.48	249.93	1779540.33	27.46	71	PASCL
	14.	550	40	977	276	2367	19192.88	278.81	1321202.65	20.39	41	OS.M
	15.	593	90	917	242	2922	23138.92	222.79	2403240.16	37.09	93	SPELL
*	16.	658	40	1069	268	4334	34898.21	289.95	4287760.88	66.17	75	ORGZN
	17.	677	90	1083	269	3601	29065.34	243.48	3469642.88	53.54	81	OS.LW
	18.	720	60	841	289	4238	34645.33	274.78	4368176.57	67.41	88	PRABU
	19.	740	60	987	315	4162	34541.30	245.94	4851077.50	74.86	124	MARXL
*	20.	823	80	1077	292	4371	35995.82	241.94	5411633.70	83.51	142	MTRXF
	21.	879	70	1566	557	5598	51062.34	668.65	3899431.54	60.18	97	OS.HL
	22.	1139	60	1471	562	6260	57181.51	563.89	5798534.75	89.48	112	PRABU
	23.	1311	160	1579	436	6505	57037.04	415.70	7825830.00	120.77	231	REVEW
	24.	1653	200	2579	619	10099	93656.06	608.27	14420350.55	222.54	179	PASER
	25.	1786	225	3734	738	10355	98657.02	740.63	13141825.43	202.81	253	NCRCN
	26.	1821	250	2332	626	10267	95380.62	628.70	14470253.18	223.31	268	CMP.L
	27.	2165	275	3742	965	12560	124524.68	1076.20	14408531.67	222.35	361	STEVE
	28.	2683	320	4068	1163	16655	169608.45	1283.73	22409023.30	345.82	325	CMP.C

where

DSI: the number of source line of instructions excluding comments

SLOC: the number of total source lines including comments

ACT: actual development effort

n: Halstead's vocabulary size (i.e. total number of distinct operators and operands)

N: Halstead's program length (i.e. total number of occurrences of operators and operands)

V: Halstead's program volume

V^*: Halstead's potential volume

E: Halstead's total number of elementary mental discriminations required to generate a program (i.e. program effort)

T: Halstead's program effort in terms of hours

V(G): McCabe's cyclomatic complexity (i.e. program control paths) which is equal to the number of predicates plus one

CLASS: source of program

'*': Values of each matric parameter listed in above lines are the averages of projects collected in each individual class. Detailed line numbers are listed below:

line 1: mean metrics of line 1, 3, and 7 of Table 6.
line 5: mean metrics of line 5, 6, 9, 10, 12, and 24 of Table 6.
line 6: mean metrics of line 13, 14, and 18 of Table 6.
line 7: mean metrics of line 8, 17, 18, and 25 of Table 6.
line 8: mean metrics of line 15, 22, 26, and 27 of Table 6.
line 10: mean metrics of line 16, 29, 31, and 34 of Table 6.
line 13: mean metrics of line 21, 30, 39, and 40 of Table 6.
line 16: mean metrics of line 38 and 41 of Table 6.
line 20: mean metrics of line 36, 37,44, and 47 of Table 6.

Table 9. Calculated Technology Constant and Technology Constant Derived by Putnam for 28 FIU Projects

NBR	DSI	T_d	C		C_k	ESTM
1	180	9	84381	#	84381	22.50
2	180	20	22298		28657	23.55
3	194	4	351355	#	351355	10.00
4	224	30	14117		17711	37.98
5	248	15	49622	#	49622	37.50
6	296	18	43706		46368	37.69
7	316	22	33396		35422	46.09
8	384	80	4719		5168	152.32
9	392	40	15295		17711	64.41
10	473	80	5813		6765	126.91
11	480	50	12912		13530	108.64
12	535	30	33718		35422	64.69
13	547	20	67761	#	67761	50.00
14	550	40	21460		21892	94.20
15	593	90	5989		6765	156.12
16	658	40	25674		28657	71.92
17	677	90	6837		8362	123.01
18	720	60	14293		17711	78.83
19	740	60	14690		17711	85.59
20	823	80	10144		10946	157.82
21	879	70	13495		13530	173.68
22	1139	60	22610		28657	73.68
23	1311	160	5075		5168	378.82
24	1653	200	4411		5168	311.03
25	1786	225	3917		4181	462.54
26	1821	250	3350		4181	321.66
27	2165	275	3398		4181	369.21
28	2638	320	3271		4181	383.26

where

DSI: the number of source line of instructions excluding comments

T_d: the development time in hours

C: technology constant comupted by using the equation developed by Putnam for simple software cost estimation system. It defines as

$$C = S / (E / B)^{1/3} (T_d)^{4/3}$$

C_k: technology constant derived by Putnam

ESTM: estimated development effort (hours) computed from Putnam SLIM's model

#: represents the value of C_k beyonds the range of original technology constant derived by Putnam (see table 2)

Table 10. Ratings of the Organic Mode of Boehm's Intermediate COCOMO Model for 28 FIU Projects

NBR	DSI	CLASS	RL	DT	CP	TM	ST	VR	TR	AC	AE	PC	VE	LE	MD	TL	SC	ESTM
1	180	DS.KK	VL	N	VL	N	N	L	L	N	VL	N	N	N	N	N	N	13.00
2	180	DS.LF	VL	N	VL	N	N	L	L	N	VL	N	N	N	N	N	N	13.00
3	194	MACKy	VL	N	VL	N	N	L	L	VH	N	VH	H	H	N	N	N	4.33
4	224	LOOK	VL	N	VL	N	N	L	L	H	L	H	H	H	N	N	N	8.67
5	248	DS.KB	VL	N	VL	N	N	L	L	N	VL	N	N	N	N	N	N	17.33
6	296	DS.LG	VL	N	VL	N	N	L	L	N	VL	N	N	N	N	N	N	21.67
7	316	DS.LB	VL	N	VL	N	N	L	L	N	VL	N	N	N	N	N	N	21.67
8	384	OS.BK	VL	N	VL	N	N	L	L	H	VL	H	N	H	N	N	N	17.33
9	392	OS.LI	VL	N	VL	N	N	L	L	H	VL	H	H	H	N	N	N	17.33
10	473	SFENG	VL	N	VL	N	N	L	L	H	VL	H	H	H	N	N	N	21.67
11	480	UPDTE	VL	N	VL	N	N	L	L	H	L	H	H	H	N	N	N	17.33
12	535	PRABU	VL	N	VL	N	N	L	L	VH	L	VH	H	H	N	N	N	13.00
13	547	PASCL	VL	N	VL	N	N	L	L	L	VL	L	L	L	N	N	N	60.67
14	550	OS.MK	VL	N	VL	N	N	L	L	H	VL	H	N	H	N	N	N	26.00
15	593	SPELL	VL	N	VL	N	N	L	L	N	VL	N	N	N	N	N	N	43.33
16	658	ORGZN	VL	N	VL	N	N	L	L	N	VL	N	N	N	N	N	N	47.67
17	677	OS.LW	VL	N	VL	N	N	L	L	N	VL	N	H	H	N	N	N	39.00
18	720	PRABU	VL	N	VL	N	N	L	L	VH	L	VH	H	H	N	N	N	17.33
19	740	MARXL	VL	N	VL	N	N	L	L	VH	VL	VH	H	H	N	N	N	21.67
20	823	MTRXF	VL	N	VL	N	N	L	L	H	VL	H	H	H	N	N	N	39.00
21	879	OS.HL	VL	N	VL	N	N	L	L	H	VL	H	H	H	N	N	N	39.00
22	1139	PRABU	VL	N	VL	N	N	L	L	VH	L	VH	H	H	N	N	N	30.33
23	1311	REVEW	VL	N	VL	N	N	L	L	H	L	H	H	H	N	N	N	52.00
24	1653	PASER	VL	N	VL	N	N	L	L	H	VL	H	H	H	N	N	N	78.00
25	1786	NCRCN	VL	N	VL	N	N	L	L	VH	L	VH	H	H	N	N	N	47.67
26	1821	CMP.L	VL	N	VL	N	N	L	L	H	VL	H	H	H	N	N	N	86.67
27	2165	STEVE	VL	N	VL	N	N	L	L	H	VL	H	H	H	N	N	N	104.00
28	2683	CMP.C	VL	N	VL	N	N	L	L	H	VL	H	H	H	N	N	N	130.00

where

DSI: the number of source line of instructions excluding comments
CLASS: source of program
RL: required software reliability
DT: data base size
CP: product complexity
TM: execution time constraint
ST: main storage constraint
VR: virtual machine volatility
TR: computer turnaround time
AC: analyst capability
AE: application experience
PC: programmer capability
VE: virtual machine experience
LE: programming language experience
MD: modern programming practices
TL: use of software tools

SC: required development schedule
ESTM: estimated development effort(hours) computed from the organic
mode of Boehm's Intermediate COCOMO model.
VL: very low
L: low
N: nominal
H: high
VH: very high

Table 11. Ratings of Six Effort Multipliers of Adjusted COCOMO Model for 28 FIU Projects

DSI	ACT	CLASS	AEXP	PCAP	VEXP	LEXP	TURN	MM	HRS	TYP	ESTM
180	20	DS.KK	VL	N	N	N	L	0.6	26.00	VL	18.20
180	9	DS.LF	VL	N	N	N	L	0.6	26.00	VL	18.20
194	4	MACKY	N	VH	H	H	L	0.3	13.00	N	13.00
224	30	LOOK	L	H	H	H	L	0.5	21.67	VH	28.17
248	15	DS.KB	VL	N	N	N	L	0.8	34.67	VL	24.27
296	18	DS.LG	VL	N	N	N	L	1.0	43.33	VL	30.33
316	22	DS.LB	VL	N	N	N	L	1.0	43.33	VH	56.33
384	80	OS.BK	VL	H	N	H	L	1.1	47.67	VH	61.97
392	40	OS.LI	VL	H	H	H	L	1.0	43.33	VH	56.33
473	80	SFENG	VL	H	H	H	L	1.2	52.00	VH	67.60
480	50	UPDTE	L	H	H	H	L	1.0	43.33	VL	30.33
535	30	PRABU	L	VH	H	H	L	1.0	43.33	VL	30.33
547	20	PASCL	VL	L	L	L	L	2.5	108.33	VL	75.83
550	40	OS.MK	VL	H	N	H	L	1.5	65.00	VH	84.50
593	90	SPELL	VL	N	N	N	L	2.0	86.67	VH	112.67
658	40	ORGZN	VL	N	N	N	L	2.2	95.33	VL	66.73
677	90	OS.LW	VL	N	H	H	L	2.0	86.67	VH	112.67
720	60	PRABU	L	VH	H	H	L	1.3	56.33	VL	39.43
740	60	MARXL	VL	VH	H	H	L	1.6	69.33	L	58.93
823	80	MTRXF	VL	H	H	H	L	2.1	91.00	L	77.35
879	70	OS.HL	VL	H	H	H	L	2.3	99.67	VH	129.57
1139	60	PRABU	L	VH	H	H	L	2.2	95.33	VL	66.73
1311	160	REVEW	L	H	H	H	L	3.0	130.00	VH	169.00
1653	200	PASER	VL	H	H	H	L	4.4	190.67	N	190.67
1786	225	NCRCN	L	VH	H	H	L	3.5	151.67	N	151.67
1821	250	CMP.L	VL	H	H	H	L	4.9	212.33	N	212.33
2165	275	STEVE	VL	H	H	H	L	5.8	251.33	N	251.33
2683	320	CMP.C	VL	H	H	H	L	7.3	316.73	N	316.73

where

DSI: the number of source line of instructions excluding comments

ACT: actual development effort

CLASS: source of program

AEXP: applications experience

PCAP: programmer capability

VEXP: virtual machine experience

LEXP: programming language experience

TURN: computer turnaround time

MM: estimated development effort in terms of man-month computed by WICOMO without including the attribute of application type.

HRS: same as 'MM' except that the development effort was defined in hours

TYP: project type

ESTM: estimated development effort (hours) computed by the adjusted COCOMO model using only six effort mutipliers: AEXP, PCAP, VEXP, LEXP, TURN and application type.

VL: very low
L: low
N: nominal
H: high
VH: very high

Table 12. Complexity metrics of 324 NCR programs

NBR	DSI	SLOC	n	N	V	V(G)	DSI/V(G)
1.	603.	833.	334.	3107.	26048.17	87.	6.93
2.	314.	376.	298.	1633.	13421.90	41.	7.66
3.	1434.	1877.	805.	7727.	74587.53	206.	6.96
4.	22.	43.	48.	107.	597.59	2.	11.00
5.	104.	141.	128.	600.	4200.00	12.	8.67
6.	259.	301.	181.	1545.	11587.26	45.	5.76
7.	790.	1155.	312.	3638.	30142.29	144.	5.49
8.	50.	73.	80.	260.	1643.70	3.	16.67
9.	387.	592.	231.	1613.	12664.87	40.	9.67
10.	241.	491.	160.	948.	6941.19	6.	40.17
11.	1659.	2177.	553.	7653.	69727.52	193.	8.60
12.	183.	190.	127.	1004.	7016.64	36.	5.08
13.	249.	289.	153.	1343.	9746.67	46.	5.41
14.	572.	843.	240.	2628.	20779.31	123.	4.65
15.	243.	421.	212.	1425.	11012.29	30.	8.10
16.	1122.	1620.	610.	7468.	69098.90	231.	4.86
17.	786.	1070.	528.	4304.	38927.07	105.	7.49
18.	59.	81.	86.	261.	1677.26	4.	14.75
19.	32.	51.	52.	152.	866.47	5.	6.40
20.	1172.	1837.	670.	7444.	69884.40	230.	5.10
21.	39.	60.	64.	163.	978.00	3.	13.00
22.	336.	458.	161.	1755.	12865.76	60.	5.60
23.	1482.	1725.	599.	6794.	62684.24	191.	7.76
24.	1038.	1248.	631.	4777.	44433.25	75.	13.84
25.	841.	1072.	569.	4024.	36828.79	64.	13.14
26.	100.	159.	100.	571.	3793.64	12.	8.33
27.	65.	72.	84.	305.	1949.66	7.	9.29
28.	83.	99.	97.	376.	2481.57	9.	9.22
29.	161.	169.	127.	805.	5625.89	26.	6.19
30.	635.	899.	375.	3247.	27764.27	77.	8.25
31.	315.	415.	246.	1512.	12009.08	17.	18.53
32.	713.	861.	426.	3785.	33060.88	69.	10.33
33.	76.	141.	69.	336.	2052.46	10.	7.60
34.	152.	223.	141.	718.	5126.20	7.	21.71
35.	160.	212.	154.	921.	6692.71	20.	8.00
36.	60.	101.	67.	223.	1352.74	9.	6.67
37.	1213.	1491.	654.	5851.	54725.26	207.	5.86
38.	113.	172.	93.	508.	3321.89	31.	3.65
39.	34.	85.	61.	138.	818.44	3.	11.33
40.	93.	132.	99.	384.	2545.67	12.	7.75
41.	175.	241.	173.	895.	6653.99	22.	7.95
42.	227.	294.	193.	1131.	8587.07	32.	7.09
43.	34.	96.	61.	138.	818.44	3.	11.33
44.	107.	184.	109.	582.	3939.08	12.	8.92
45.	119.	170.	85.	564.	3614.90	16.	7.44
46.	40.	64.	64.	181.	1086.00	7.	5.71
47.	40.	80.	57.	182.	1061.59	5.	8.00
48.	28.	44.	57.	137.	799.11	2.	14.00

49.	32.	48.	65.	152.	915.40	3.	10.67
50.	64.	70.	73.	324.	2005.50	8.	8.00
51.	112.	164.	114.	574.	3922.08	15.	7.47
52.	125.	181.	105.	668.	4485.12	17.	7.35
53.	76.	115.	95.	376.	2470.27	6.	12.67
54.	438.	606.	163.	2157.	15851.21	72.	6.08
55.	160.	220.	121.	825.	5708.06	23.	6.96
56.	157.	239.	113.	769.	5244.72	28.	5.61
57.	374.	577.	288.	1807.	14763.05	30.	12.47
58.	112.	168.	108.	605.	4086.71	17.	6.59
59.	624.	892.	317.	3281.	27259.66	76.	8.21
60.	128.	188.	115.	675.	4620.71	20.	6.40
61.	50.	93.	79.	266.	1676.81	6.	8.33
62.	122.	162.	100.	618.	4105.90	16.	7.63
63.	530.	688.	472.	2626.	23325.82	29.	18.28
64.	223.	474.	203.	1122.	8600.51	15.	14.87
65.	314.	418.	290.	1551.	12687.04	30.	10.47
66.	583.	844.	505.	2926.	26275.89	31.	18.81
67.	485.	664.	416.	2366.	20585.24	23.	21.09
68.	370.	478.	284.	1711.	13944.22	23.	16.09
69.	89.	147.	88.	447.	2887.37	15.	5.93
70.	189.	299.	141.	1020.	7282.34	22.	8.59
71.	96.	179.	113.	492.	3355.53	3.	32.00
72.	123.	180.	110.	659.	4468.92	15.	8.20
73.	32.	39.	52.	145.	826.56	1.	32.00
74.	113.	172.	93.	493.	3223.81	12.	9.42
75.	236.	485.	131.	921.	6477.78	4.	59.00
76.	57.	101.	97.	307.	2026.17	5.	11.40
77.	56.	96.	94.	290.	1900.83	5.	11.20
78.	59.	104.	97.	314.	2072.37	6.	9.83
79.	56.	100.	94.	290.	1900.83	5.	11.20
80.	78.	128.	97.	363.	2395.77	6.	13.00
81.	301.	441.	157.	1661.	12116.36	47.	6.40
82.	47.	89.	69.	196.	1197.27	3.	15.67
83.	422.	612.	261.	1894.	15204.85	56.	7.54
84.	144.	198.	117.	757.	5200.87	22.	6.55
85.	293.	432.	339.	1525.	12817.84	8.	36.62
86.	431.	618.	403.	2151.	18616.12	29.	14.86
87.	108.	169.	133.	637.	4494.21	13.	8.31
88.	165.	230.	135.	849.	6008.22	24.	6.88
89.	101.	120.	114.	644.	4400.38	17.	5.94
90.	341.	423.	150.	1658.	11985.38	54.	6.31
91.	237.	262.	208.	1557.	11989.58	31.	7.65
92.	202.	222.	205.	1375.	10559.29	22.	9.18
93.	491.	705.	439.	2543.	22322.65	28.	17.54
94.	816.	985.	360.	3993.	33907.97	96.	8.50
95.	173.	220.	138.	819.	5821.88	29.	5.97
96.	88.	125.	114.	444.	3033.80	10.	8.80
97.	63.	110.	91.	309.	2010.91	5.	12.60
98.	104.	142.	129.	548.	3842.15	11.	9.45
99.	238.	379.	186.	1213.	9145.00	35.	6.80
100.	57.	102.	97.	307.	2026.17	5.	11.40
101.	58.	98.	94.	290.	1900.83	5.	11.60
102.	137.	237.	112.	776.	5282.51	28.	4.89

103.	78.	125.	115.	422.	2888.80	6.	13.00
104.	56.	100.	94.	290.	1900.83	5.	11.20
105.	124.	191.	106.	605.	4070.39	20.	6.20
106.	20.	74.	44.	92.	502.27	2.	10.00
107.	83.	102.	95.	394.	2588.52	13.	6.38
108.	307.	441.	215.	1541.	11939.96	40.	7.67
109.	48.	90.	73.	213.	1318.43	3.	16.00
110.	98.	146.	128.	565.	3955.00	5.	19.60
111.	478.	675.	386.	2381.	20458.64	45.	10.62
112.	134.	211.	106.	705.	4743.18	18.	7.44
113.	140.	216.	118.	774.	5327.17	21.	6.67
114.	160.	241.	110.	830.	5628.53	25.	6.40
115.	100.	171.	107.	551.	3714.55	11.	9.09
116.	1057.	1483.	587.	5800.	53343.86	148.	7.14
117.	77.	117.	94.	375.	2457.97	10.	7.70
118.	354.	487.	293.	1726.	14144.15	45.	7.87
119.	142.	192.	129.	757.	5307.50	22.	6.45
120.	55.	74.	87.	301.	1939.33	4.	13.75
121.	401.	560.	330.	2420.	20246.50	72.	5.57
122.	463.	667.	411.	2297.	19944.84	24.	19.29
123.	262.	315.	191.	1240.	9396.01	39.	6.72
124.	105.	160.	95.	508.	3337.49	13.	8.08
125.	222.	320.	140.	1084.	7728.14	33.	6.73
126.	289.	296.	193.	1496.	11358.32	41.	7.05
127.	290.	311.	190.	1533.	11604.59	43.	6.74
128.	386.	569.	273.	1836.	14858.30	40.	9.65
129.	116.	133.	119.	552.	3805.94	12.	9.67
130.	116.	134.	113.	544.	3710.18	13.	8.92
131.	445.	587.	307.	2135.	17639.57	35.	12.71
132.	211.	329.	277.	1029.	8349.04	4.	52.75
133.	113.	164.	129.	542.	3800.09	10.	11.30
134.	130.	199.	116.	694.	4759.44	16.	8.13
135.	592.	858.	312.	2840.	23530.54	69.	8.58
136.	251.	500.	153.	1056.	7663.80	8.	31.37
137.	955.	1321.	584.	4526.	41593.15	80.	11.94
138.	1528.	2153.	703.	6881.	65076.24	156.	9.79
139.	51.	62.	74.	257.	1595.83	7.	7.29
140.	98.	160.	104.	450.	3015.20	8.	12.25
141.	588.	870.	341.	2766.	23272.09	64.	9.19
142.	1434.	1959.	657.	6368.	59602.88	148.	9.69
143.	609.	874.	374.	2889.	24691.98	54.	11.28
144.	98.	157.	111.	455.	3091.46	10.	9.80
145.	19.	36.	44.	88.	480.43	2.	9.50
146.	29.	44.	48.	129.	720.46	4.	7.25
147.	29.	49.	55.	129.	745.80	3.	9.67
148.	130.	213.	112.	658.	4479.24	18.	7.22
149.	32.	45.	54.	164.	943.80	6.	5.33
150.	1426.	1999.	1025.	8724.	87252.28	49.	29.10
151.	66.	144.	61.	292.	1731.78	7.	9.43
152.	71.	77.	78.	323.	2030.18	9.	7.89
153.	36.	82.	63.	153.	914.52	3.	12.00
154.	34.	93.	57.	156.	909.93	2.	17.00
155.	315.	421.	152.	1604.	11625.68	49.	6.43
156.	112.	175.	121.	527.	3646.24	12.	9.33

157.	730.	1088.	466.	3337.	29579.79	54.	13.52
158.	1131.	1763.	659.	5402.	50585.05	91.	12.43
159.	674.	982.	498.	3281.	29397.77	50.	13.48
160.	617.	819.	366.	3112.	26500.86	73.	8.45
161.	20.	40.	41.	86.	460.75	2.	10.00
162.	703.	1093.	401.	3904.	33759.68	96.	7.32
163.	448.	611.	304.	2338.	19283.65	42.	10.67
164.	184.	241.	158.	847.	6186.30	18.	10.22
165.	276.	436.	220.	1424.	11080.66	18.	15.33
166.	97.	160.	101.	554.	3688.65	11.	8.82
167.	40.	47.	65.	199.	1198.45	7.	5.71
168.	132.	141.	102.	625.	4170.27	19.	6.95
169.	26.	45.	59.	135.	794.16	4.	6.50
170.	593.	768.	501.	3145.	28206.46	33.	17.97
171.	100.	148.	111.	608.	4131.00	12.	8.33
172.	88.	106.	104.	519.	3477.53	13.	6.77
173.	32.	39.	68.	160.	973.99	3.	10.67
174.	52.	67.	84.	216.	1380.74	4.	13.00
175.	301.	433.	146.	1508.	10842.26	51.	5.90
176.	248.	325.	114.	1259.	8602.61	42.	5.90
177.	70.	111.	101.	389.	2590.04	6.	11.67
178.	346.	489.	245.	2118.	16809.80	52.	6.65
179.	380.	524.	257.	2309.	18484.99	56.	6.79
180.	1168.	1639.	591.	5257.	48401.27	153.	7.63
181.	768.	1108.	572.	3929.	35989.13	55.	13.96
182.	295.	405.	158.	1543.	11269.73	34.	8.68
183.	141.	221.	148.	816.	5882.91	20.	7.05
184.	246.	497.	162.	1060.	7780.24	7.	35.14
185.	390.	541.	340.	1878.	15792.84	13.	30.00
186.	18.	32.	44.	85.	464.05	2.	9.00
187.	50.	65.	76.	238.	1487.01	8.	6.25
188.	136.	201.	116.	731.	5013.18	18.	7.56
189.	265.	516.	183.	1092.	8207.14	4.	66.25
190.	1716.	2423.	697.	8424.	79564.80	198.	8.67
191.	78.	97.	85.	407.	2608.62	15.	5.20
192.	27.	58.	50.	111.	626.47	1.	27.00
193.	26.	57.	48.	107.	597.59	1.	26.00
194.	955.	1143.	214.	3587.	27768.64	181.	5.28
195.	175.	249.	116.	912.	6254.48	27.	6.48
196.	148.	208.	116.	752.	5157.20	20.	7.40
197.	126.	192.	101.	673.	4480.98	17.	7.41
198.	124.	164.	115.	850.	5818.67	10.	12.40
199.	36.	53.	62.	203.	1208.70	5.	7.20
200.	312.	430.	286.	1918.	15650.63	21.	14.86
201.	1470.	1914.	603.	7842.	72428.82	215.	6.84
202.	623.	741.	290.	4055.	33169.53	101.	6.17
203.	1637.	2280.	621.	8074.	74914.20	287.	5.70
204.	535.	585.	300.	2995.	24645.31	112.	4.78
205.	138.	196.	139.	694.	4940.55	18.	7.67
206.	305.	367.	219.	1608.	12501.86	52.	5.87
207.	558.	656.	280.	3120.	25363.36	121.	4.61
208.	424.	491.	256.	2539.	20312.00	91.	4.66
209.	533.	640.	291.	3104.	25405.85	121.	4.40
210.	285.	348.	200.	1534.	11725.68	49.	5.82

211.	236.	325.	139.	1212.	8628.16	43.	5.49
212.	317.	370.	231.	1685.	13230.20	53.	5.98
213.	200.	261.	141.	1031.	7360.88	33.	6.06
214.	247.	338.	177.	1221.	9117.95	30.	8.23
215.	126.	160.	153.	658.	4775.36	8.	15.75
216.	315.	411.	213.	1618.	12514.76	48.	6.56
217.	469.	621.	440.	2474.	21725.08	20.	23.45
218.	55.	89.	73.	240.	1485.56	4.	13.75
219.	1106.	1505.	656.	5956.	55733.58	182.	6.08
220.	50.	78.	90.	310.	2012.47	1.	50.00
221.	320.	503.	320.	1912.	15911.53	20.	16.00
222.	146.	191.	143.	793.	5677.78	13.	11.23
223.	215.	355.	153.	1146.	8316.97	24.	8.96
224.	137.	256.	119.	626.	4316.16	13.	10.54
225.	388.	581.	399.	1895.	16373.26	15.	25.87
226.	572.	662.	313.	3090.	25616.16	106.	5.40
227.	319.	406.	276.	1570.	12730.38	29.	11.00
228.	109.	160.	106.	601.	4043.48	13.	8.38
229.	41.	57.	59.	204.	1200.06	5.	8.20
230.	182.	235.	162.	945.	6936.16	27.	6.74
231.	94.	139.	102.	483.	3222.78	10.	9.40
232.	41.	61.	67.	209.	1267.81	2.	20.50
233.	152.	200.	116.	748.	5129.77	25.	6.08
234.	1173.	1616.	653.	6286.	58780.00	155.	7.57
235.	1331.	1676.	608.	6638.	61387.74	138.	9.64
236.	163.	241.	181.	754.	5654.88	4.	40.75
237.	73.	98.	97.	342.	2257.17	6.	12.17
238.	244.	495.	157.	931.	6791.29	5.	48.80
239.	711.	936.	345.	4040.	34059.03	118.	6.03
240.	43.	50.	69.	196.	1197.27	3.	14.33
241.	1571.	2163.	771.	7312.	70126.37	213.	7.38
242.	85.	179.	114.	535.	3655.60	10.	8.50
243.	979.	1591.	649.	6027.	56304.68	118.	8.30
244.	30.	51.	58.	121.	708.82	1.	30.00
245.	171.	272.	139.	892.	6350.10	24.	7.13
246.	26.	69.	59.	120.	705.92	3.	8.67
247.	47.	65.	74.	197.	1223.26	2.	23.50
248.	68.	108.	83.	327.	2084.64	8.	8.50
249.	1061.	1500.	561.	5496.	50188.69	100.	10.61
250.	139.	189.	128.	704.	4928.00	10.	13.90
251.	52.	88.	87.	247.	1591.41	6.	8.67
252.	52.	97.	87.	247.	1591.41	6.	8.67
253.	177.	295.	129.	968.	6786.87	30.	5.90
254.	270.	387.	231.	1370.	10756.90	15.	18.00
255.	52.	70.	77.	268.	1679.50	4.	13.00
256.	72.	113.	102.	395.	2635.61	6.	12.00
257.	400.	522.	220.	1994.	15516.03	58.	6.90
258.	510.	684.	259.	2487.	19937.80	55.	9.27
259.	93.	136.	95.	487.	3199.52	10.	9.30
260.	77.	133.	86.	415.	2666.90	14.	5.50
261.	148.	214.	100.	736.	4889.88	21.	7.05
262.	347.	499.	320.	1806.	15029.40	28.	12.39
263.	3631.	4960.	1186.	17019.	173796.12	449.	8.09
264.	432.	592.	237.	2203.	17378.90	62.	6.97

265.	100.	144.	110.	535.	3628.03	12.	8.33
266.	126.	179.	107.	690.	4651.61	16.	7.88
267.	42.	78.	78.	224.	1407.93	5.	8.40
268.	256.	506.	164.	970.	7136.83	5.	51.20
269.	1770.	2427.	513.	8023.	72229.58	248.	7.14
270.	1928.	2252.	573.	8753.	80198.41	292.	6.60
271.	196.	345.	184.	943.	7094.72	10.	19.60
272.	125.	172.	100.	616.	4092.62	17.	7.35
273.	81.	128.	102.	391.	2608.92	7.	11.57
274.	119.	165.	111.	597.	4056.27	16.	7.44
275.	137.	225.	133.	723.	5100.97	6.	22.83
276.	30.	74.	55.	147.	849.86	1.	30.00
277.	96.	137.	117.	422.	2899.29	8.	12.00
278.	118.	151.	124.	589.	4096.02	15.	7.87
279.	89.	115.	105.	433.	2907.27	12.	7.42
280.	97.	134.	73.	467.	2890.65	22.	4.41
281.	92.	107.	96.	468.	3081.76	13.	7.08
282.	93.	109.	99.	485.	3215.24	12.	7.75
283.	272.	399.	187.	1292.	9750.59	35.	7.77
284.	450.	543.	257.	2499.	20006.06	88.	5.11
285.	321.	459.	235.	1928.	15185.92	35.	9.17
286.	1204.	1744.	748.	6915.	66016.77	168.	7.17
287.	57.	106.	97.	345.	2276.97	4.	14.25
288.	46.	113.	72.	222.	1369.72	4.	11.50
289.	27.	44.	56.	124.	720.11	1.	27.00
290.	373.	481.	365.	2106.	17925.75	19.	19.63
291.	217.	302.	137.	1210.	8588.62	20.	10.85
292.	191.	264.	193.	2248.	17067.84	18.	10.61
293.	577.	801.	483.	2726.	24304.69	35.	16.49
294.	1277.	1874.	659.	6375.	59696.36	155.	8.24
295.	486.	689.	232.	2381.	18709.85	52.	9.35
296.	284.	412.	233.	1756.	13809.51	34.	8.35
297.	208.	400.	154.	1052.	7644.66	32.	6.50
298.	1044.	1624.	616.	5466.	50652.25	75.	13.92
299.	554.	733.	757.	2448.	23413.04	25.	22.16
300.	656.	814.	808.	3141.	30336.44	35.	18.74
301.	430.	687.	346.	2058.	17358.46	23.	18.70
302.	1502.	2586.	599.	8472.	78166.16	213.	7.05
303.	522.	696.	417.	2925.	25458.92	79.	6.61
304.	1040.	1568.	549.	5123.	46622.69	110.	9.45
305.	91.	118.	102.	420.	2802.42	9.	10.11
306.	2796.	4238.	1069.	15028.	151212.43	347.	8.06
307.	356.	411.	291.	1741.	14249.87	29.	12.28
308.	442.	625.	530.	2157.	19520.52	25.	17.68
309.	920.	1021.	866.	4057.	39589.11	70.	13.14
310.	434.	648.	455.	1993.	17597.64	27.	16.07
311.	967.	1067.	891.	4375.	42871.86	66.	14.65
312.	35.	51.	60.	197.	1163.66	4.	8.75
313.	1685.	2231.	1249.	8107.	83393.12	143.	11.78
314.	95.	145.	118.	490.	3372.50	8.	11.88
315.	315.	633.	197.	1827.	13925.49	20.	15.75
316.	363.	445.	294.	2372.	19449.62	25.	14.52
317.	1164.	1614.	415.	5460.	47485.44	148.	7.86
318.	680.	1256.	474.	3656.	32497.24	44.	15.45

319.	102.	207.	102.	653.	4357.09	9.	11.33
320.	773.	1190.	526.	4029.	36417.80	63.	12.27
321.	827.	1209.	387.	4557.	39172.84	106.	7.80
322.	605.	824.	449.	3312.	29180.61	55.	11.00
323.	175.	286.	169.	1113.	8237.18	17.	10.29
324.	260.	329.	263.	2792.	22444.66	48.	5.42

---

where

DSI: the number of source line of instructions excluding comments

SLOC: the number of source lines including comments.

n: Halstead's vocabulary (i.e. total number of distinct operators and operands)

N: Halstead's program length (i.e. total number of occurrences of operators and operands)

V: Halstead's program volume

V(G): McCabe's cyclomatic complexity (i.e. basic program control paths) which is equal to the number of predicates plus one

DSI/V(G): the number of sequential instructions per V(G)

Table 13. Actual Development Effort and Calculated Model Development Efforts for 28 FIU Projects

Nbr	DSI	SLOC	ACT EFF	HLSTD EFF	IBM EFF	SLIM EFF	COCOMO EFF	ADJST COCOMOEFF	CLASS
1	180	304	20	5.74	76.25	23.55	13.00	18.20	DS.KK
2	180	309	9	5.59	77.39	22.50	13.00	18.20	DS.LF
3	194	261	4	5.58	66.37	10.00	4.33	13.00	MACKY
4	224	322	30	7.25	80.35	37.98	8.67	28.17	LOOK
5	248	350	15	16.66	86.68	37.50	17.33	24.27	DS.KB
6	296	382	18	12.13	93.86	37.69	21.67	30.33	DS.LG
7	316	524	22	13.71	125.15	46.09	21.67	56.33	DS.LB
8	384	533	80	13.46	127.10	152.32	17.33	61.97	OS.BK
9	392	679	40	18.41	158.43	64.41	17.33	56.33	OS.LI
10	473	907	80	23.16	206.18	126.91	21.67	67.60	SFENG
11	480	592	50	40.28	139.84	108.64	17.33	30.33	UPDAT
12	535	668	30	16.13	156.09	64.69	13.00	30.33	PRABU
13	547	637	20	27.46	149.48	50.00	60.67	75.83	PASCL
14	550	977	40	20.39	220.61	94.20	26.00	84.50	OS.M
15	593	917	90	37.09	208.25	156.12	43.33	112.67	SPELL
16	658	1069	40	66.17	239.44	71.92	47.67	66.73	ORGZN
17	677	1083	90	53.54	242.29	123.01	39.00	112.67	OS.LW
18	720	841	60	67.41	192.48	78.83	17.33	39.43	PRABU
19	740	987	60	74.86	222.67	85.59	21.67	58.93	MTRXL
20	823	1077	80	83.51	241.07	157.82	39.00	77.35	MTRXF
21	879	1566	70	60.18	338.91	173.68	39.00	129.57	OS.HL
22	1139	1471	60	89.48	320.15	73.68	30.33	66.73	PRABU
23	1311	1579	160	120.77	341.47	378.82	52.00	169.00	REVEW
24	1653	2579	200	222.54	533.64	311.03	78.00	190.67	PASER
25	1786	3734	225	202.81	747.32	462.54	47.67	151.67	NCRCN
26	1821	2332	250	223.31	486.92	321.66	86.67	212.33	CMP.L
27	2165	3742	275	222.35	748.77	369.21	104.00	251.33	STEVE
28	2683	4068	320	345.82	807.91	383.26	130.00	316.73	CPM.C

where

DSI: the number of source line of instructions excluding comments

SLOC: the number of source lines including comments

ACTEFF: actual development effort (hours)

HLSTDEFF: development effort computed from Halstead's Software Science (hours)

IBMEFF: development effort computed from IBM model (hours)

SLIMEFF: development effort computed from Putnam's SLIM model (hours)

COCOMOEFF: development effort computed from the Organic mode of Boehm's Intermediate COCOMO model (hours)

ADJST-COCOMOEFF: development effort computed from adjusted COCOMO model

CLASS: source of program

Table 14. Relative Errors of Models' Development Efforts of FIU Projects

NBR	X-Y1	ABS(X-Y1) / X	X-Y2	ABS(X-Y2) / X	X-Y3	ABS(X-Y3) / X	X-Y4	ABS(X-Y4) / X	X-Y5	ABS(X-Y5) / X
1	14.26	0.71	-56.25	2.81	-3.55	0.18	7.00	0.35	1.80	0.09
2	3.41	0.40	-68.39	7.60	-13.50	1.50	-4.00	0.44	-9.20	1.02
3	-1.58	0.40	-62.37	15.59	-6.00	1.50	-0.33	0.08	-9.00	2.25
4	22.75	0.76	-50.35	1.68	-7.98	0.27	21.33	0.71	1.83	0.06
5	-1.66	0.11	-71.68	4.78	-22.50	1.50	-2.33	0.16	-9.27	0.62
6	5.87	0.33	-75.86	4.21	-19.69	1.09	-3.67	0.20	-12.33	0.69
7	8.29	0.38	-103.15	4.69	-24.09	1.10	0.33	0.02	-34.33	1.56
8	66.54	0.83	-47.10	0.59	-72.32	0.91	62.67	0.78	18.03	0.23
9	21.59	0.54	-118.43	2.96	-24.41	0.61	22.67	0.57	-16.33	0.41
10	56.84	0.71	-126.18	1.58	-46.91	0.59	58.33	0.73	12.40	0.16
11	9.72	0.19	-89.84	1.80	-58.64	1.17	32.67	0.65	19.67	0.40
12	13.87	0.46	-126.09	4.20	-34.69	1.16	17.00	0.57	-0.33	0.01
13	-7.46	0.37	-129.48	6.47	-30.00	1.50	-40.67	2.03	-55.83	2.79
14	19.61	0.49	-180.61	4.52	-54.20	1.36	14.00	0.35	-44.50	1.11
15	52.91	0.59	-118.25	1.31	-66.12	0.74	46.67	0.52	-22.67	0.25
16	-26.17	0.65	-199.44	4.99	-31.92	0.80	-7.67	0.19	-26.73	0.67
17	36.46	0.41	-152.29	1.69	-33.01	0.37	51.00	0.57	-22.67	0.25
18	-7.41	0.12	-132.48	2.21	-18.83	0.31	42.67	0.71	20.57	0.34
19	-14.86	0.25	-162.67	2.71	-25.59	0.43	38.33	0.64	1.07	0.02
20	-3.51	0.04	-161.07	2.01	-77.82	0.97	41.00	0.51	2.65	0.03
21	9.82	0.14	-268.91	3.84	-103.68	1.48	31.00	0.44	-59.57	0.85
22	-29.48	0.49	-260.15	4.34	-13.68	0.23	29.67	0.50	-6.73	0.11
23	39.23	0.25	-181.47	1.13	-218.82	1.37	108.00	0.68	-9.00	0.06
24	-22.54	0.11	-333.64	1.67	-111.03	0.56	122.00	0.61	9.33	0.05
25	22.19	0.10	-522.32	2.32	-237.54	1.06	177.33	0.79	73.33	0.33
26	26.69	0.11	-236.92	0.95	-71.66	0.29	163.33	0.65	37.67	0.15
27	52.65	0.19	-473.77	1.72	-94.21	0.34	171.00	0.62	23.67	0.09
28	-25.82	0.08	-487.91	1.52	-63.26	0.20	190.00	0.59	3.27	0.01

where

X and Y are efforts of 28 projects listed in Table 13

X: represents actual development effort

Y1: represents development effort computed from Halstead's Software Science

Y2: represents development effort computed from IBM model

Y3: represents development effort computed from Putnam's SLIM model

Y4: represents development effort computed from the Organic mode of Boehm's Intermediate COCOMO model

Y5: represents development effort computed from adjusted COCOMO model

ABS: represents absolute value

ABS(X-Y1) / X: relative error of development effort computed from Halstead's Software Science

ABS(X-Y2) / X: relative error of development effort computed from IBM model

ABS(X-Y3) / X: relative error of development effort computed from

Putnam's SLIM model

ABS(X-Y4) / X: relative error of development effort computed from the Organic mode of Boehm's Intermediate COCOMO model

ABS(X-Y5) / X: relative error of development effort computed from adjusted COCOMO model

The average relative errors and their ranges of above 28 observations of all models are listed below:

MODEL	TOTAL RELATIVE ERROR	AVERAGE RELATIVE ERROR	RANGE OF RELATIVE ERROR
HALSTEAD	10.22	0.36	0.08 -- 0.83
IBM	95.89	3.42	0.59 -- 15.59
SLIM	23.59	0.84	0.18 -- 1.50
COCOMO	15.67	0.56	0.08 -- 2.03
ADJUSTED COCOMO	14.61	0.52	0.01 -- 2.79

where

AVERAGE RELATIVE ERROR: the ratio of TOTAL RELATIVE ERROR of 28 projects to 28

Table 15. Relative Errors of Models' Development Efforts of FIU Projects with Program Lengths over 1000 Lines

NBR	X-Y1	ABS(X-Y1) / X	X-Y2	ABS(X-Y2) / X	X-Y3	ABS(X-Y3) / X	X-Y4	ABS(X-Y4) / X	X-Y5	ABS(X-Y5) / X
1	-29.48	0.49	-260.15	4.34	-13.68	0.23	29.67	0.50	-6.73	0.11
2	39.23	0.25	-181.47	1.13	-218.82	1.37	108.00	0.68	-9.00	0.05
3	-22.54	0.11	-333.64	1.67	-111.03	0.56	122.00	0.61	9.33	0.05
4	22.19	0.10	-522.32	2.32	-237.54	1.06	177.33	0.79	73.33	0.33
5	26.69	0.11	-236.92	0.95	-71.66	0.29	163.33	0.65	37.67	0.15
6	52.65	0.19	-473.77	1.72	-94.21	0.34	171.00	0.63	23.67	0.09
7	-25.82	0.09	-487.91	1.52	-63.26	0.20	190.00	0.59	3.27	0.01

where

X and Y are efforts of seven projects with source instructions (DSI) more than 1000 lines

X: represents actual development effort

Y1: represents development effort computed from Halstead's Software Science

Y2: represents development effort computed from IBM model

Y3: represents development effort computed from Putnam's SLIM model

Y4: represents development effort computed from the Organic mode of Boehm's Intermediate COCOMO model

Y5: represents development effort computed from adjusted COCOMO model

ABS: represents absolute value

ABS(X-Y1) / X: relative errror of development effort computed from Halstead's Software Science

ABS(X-Y2) / X: relative error of development effort computed from IBM model

ABS(X-Y3) / X: relative error of development effort computed from Putnam's SLIM model

ABS(X-Y4) / X: relative error of development effort computed from the Organic mode of Boehm's Intermediate COCOMO model

ABS(X-Y5) / X: relative error of development effort computed from adjusted COCOMO model

The average relative errors and their ranges of above seven observations of all models are listed below:

MODEL	TOTAL RELATIVE ERROR	AVERAGE RELATIVE ERROR	RANGE OF RELATIVE ERROR
HALSTEAD	1.34	0.19	0.09 -- 0.49
IBM	13.65	1.95	0.95 -- 4.34
SLIM	4.05	0.58	0.20 -- 1.37
COCOMO	4.45	0.64	0.50 -- 0.79
ADJUSTED COCOMO	0.79	0.11	0.01 -- 0.33

where

AVERAGE RELATIVE ERROR: the ratio of TOTAL RELATIVE ERROR of seven projets to seven.

Table 16. Linear Regression Results of FIU Projecs

DEPENDENT VARIABLE	INDEPENDENT VARIABLE	R^2	r	MSE	F-STAT (0.001 SIG)	~ B_0	~ B_1	AVG SQRD DEL RESIDUAL	R^2_{pred}
ACTEFF	IBMEFF	0.8931	0.9451	849.32	217.25	-16.5486	0.3904	942.99	0.87
ACTEFF	SLIMEFF	0.8816	0.9389	941.15	193.52	-3.3760	0.6294	1,232.23	0.83
ACTEFF	COCOMOEFF	0.7845	0.8857	1,712.22	94.66	-8.3229	2.5471	1,773.28	0.76
ACTEFF	ADJSTEFF	0.9056	0.9516	749.99	249.47	-11.8439	1.0856	785.93	0.89
ACTEFF	HLSTDEFF	0.9142	0.9561	682.02	276.92	16.0050	0.9496	760.67	0.90
ACTEFF	V(G)	0.9003	0.9489	791.99	234.86	-3.3093	0.8543	892.06	0.88
ACTEFF	V^*	0.7886	0.8889	1,680.22	96.96	-8.1222	0.2633	1,766.55	0.76
n	DSI	0.9291	0.9638	3,337.63	680.89	55.3674	0.3735	3,700.86	0.92
V	DSI	0.9767	0.9883	26,825,643.34	2,183.07	-7,995.6900	59.9500	32,155,787.63	0.97
V(G)	DSI	0.9084	0.9531	730.42	515.37	-12.0897	0.1519	880.74	0.89

where

R^2 (squared correlation):
represents the proportion of the total variability of the dependent variable, Y, that is accounted for by the independent variable, X. It is defined as

$$R^2 = SSR / S_{YY}$$

SSR: represents regression sum of squares. It is defined as

$$SSR = \Sigma (\hat{Y} - \overline{Y})^2$$

S_{YY}: represents the variability in Y without considering the effect of the regressor variable X. It is defined as

$$S_{YY} = \Sigma (Y - \overline{Y})^2$$

r (correlation coefficient):
measures the strength of the relationship between variables X and Y. It is the square root of R^2

MSE (residual mean square or mean square error):
measures the average squared deviation between the observed and predicted values of dependent variable

F-STAT (F statistics):
It is used to test the linear relationship between variables X and Y (i.e. to test Ho: $B_1=0$). It is defined as

$$F = MSR / MSE$$

where
MSR : represents mean square due to regression
MSE : as described above

SIG (level of significance): represents the weight of the evidence for rejecting the null hypothesis, Ho. This probability is sometimes designated by the letter P. For example, in linear regression, if P < 0.001 SIG, it means that when Ho is true (i.e. no linear relationship), the probability of rejecting Ho is less than 1/1000 given the sample results. In other words there is a very little chance of saying that there is no linear relationship between X and Y.

$\tilde{B}_0$: represents the estimte of intercept of the regression line. It is defined as

$$\tilde{B}_0 = \bar{Y} - \tilde{B}_1 \bar{X}$$

$\tilde{B}_1$: estimates the predicted change in Y for a unit change in X. The equation is defined as

$$\tilde{B}_1 = S_{XY} / S_{XX}$$

where

$$S_{XX} = \Sigma (X)^2 - ((\Sigma X)^2 / n)$$

$$S_{XY} = \Sigma (X * Y) - (((\Sigma X) (\Sigma Y)) / n)$$

AVG SQRD DEL RESIDUAL:
represents average squared deleted residual. It is used to calculate PRESS statistics described earlier in section 5.1.4.

R^2_{pred}: represents the predicted squared multiple correlation calculating from 1 - (PRESS / S_{YY})

ACTEFF: actual development effort

IBMEFF: development effort computed from IBM model

SLIMEFF: development effort computed from Putnam's SLIM model

COCOMOEFF: development effort computed from the Organic mode of Boehm's Intermediate COCOMO model

ADJSTEFF: development effort computed from adjusted COCOMO model

HLSTDEFF: development effort computed from Halstead's Software Science

DSI: the number of source line of instructions excluding comments

n: Halstead's vocabulary (i.e. total number of occurrences of operators and operands)

V: Halstead's program volume

V^*: Halstead's potential volume of an algorithm

V(G): McCabe's cyclomatic complexity (i.e. basic program control paths) which is equal to the number of predicates plus one

Table 17. Nonlinear Regression Results of FIU Projects

DEPENDENT VARIABLE	INDEPENDENT VARIABLE	P1	P2	MSE	R^2
ACTEFF	IBMEFF	0.1366	1.1500	863.66	0.8913
ACTEFF	SLIMEFF	0.5436	1.0219	944.72	0.8811
ACTEFF	COCOMOEFF	1.5027	1.1095	1691.76	0.7870
ACTEFF	ADJUSTEFF	0.4687	1.1464	723.28	0.9090
ACTEFF	HLSTDEFF	2.3811	0.8452	726.82	0.9085
ACTEFF	V(G)	0.5356	1.0819	771.88	0.9029
ACTEFF	V^*	0.1703	1.0579	1691.50	0.7871
n	DSI	1.0700	0.8715	3882.58	0.9175
V	DSI	8.3025	1.2530	14141900.00	0.9877
V(G)	DSI	0.0769	1.0830	757.75	0.9052

where

P1,P2: parameters of the nonlinear function $Y = P1\ (X)^{P2}$

MSE, R^2: same as those described in Table 16

Table 18. Linear Regression Results of 324 NCR Programs

DEPENDENT VARIABLE	INDEPENDENT VARIABLE	R^2	r	MSE	F-STAT (0.001 SIG)	$\tilde{B}_0$	$\tilde{B}_1$	AVG SQRD DEL RESIDUAL	R^2_{pred}
n	DSI	0.8166	0.9037	8292.09	1433.74	84.4777	0.4240	8713.81	0.81
V	DSI	0.9788	0.9893	10332555.80	14862.50	-1585.2900	48.1909	10757778.27	0.98
V(G)	DSI	0.8760	0.9359	448.72	2274.41	-2.4229	0.1241	456.43	0.87

where

R^2, r, MSE, F-STAT, $\tilde{B}_0$, $\tilde{B}_1$, AVG SQRD DEL RESIDUAL, R_{pred}, n, V, V(G), and DSI are the same as those described in Table 16

Table 19. Nonlinear Regression Results of 324 NCR Programs

DEPENDENT VARIABLE	INDEPENDENT VARIABLE	P1	P2	MSE	R^2
n	DSI	5.1346	0.6744	6312.73	0.8604
V	DSI	28.2923	1.0694	10137100.00	0.9792
V(G)	DSI	0.0816	1.0560	444.74	0.8771

where

P1, P2 and R^2 are the same as those described in Table 17.

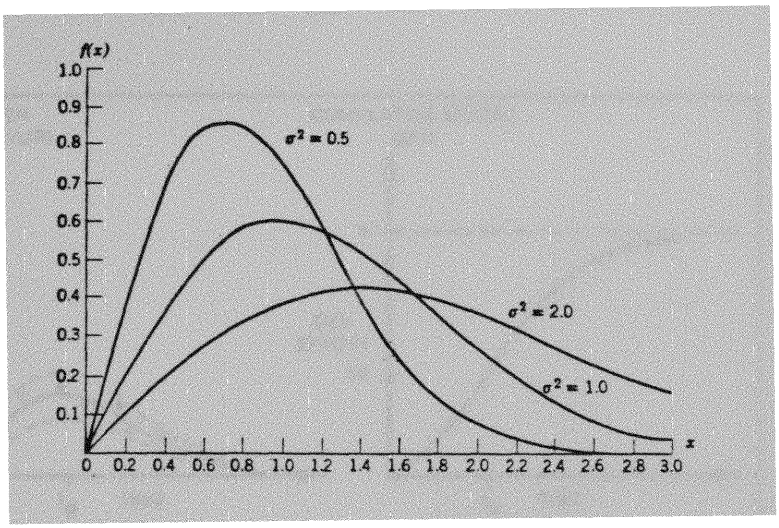


Figure 1. Rayleigh Distributions with Various Values of Variance

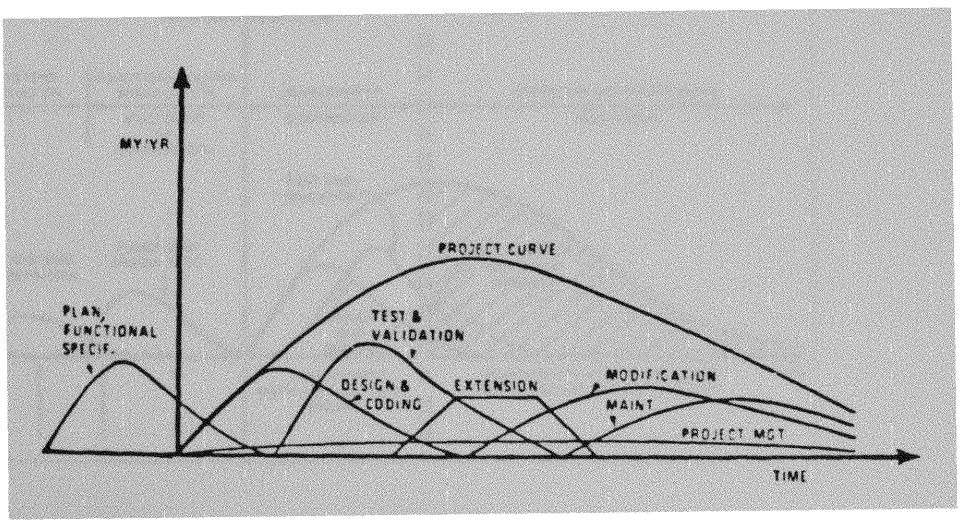


Figure 2. Pattern of Life-Cycle Effort Required to Complete a Large Scale Software Project

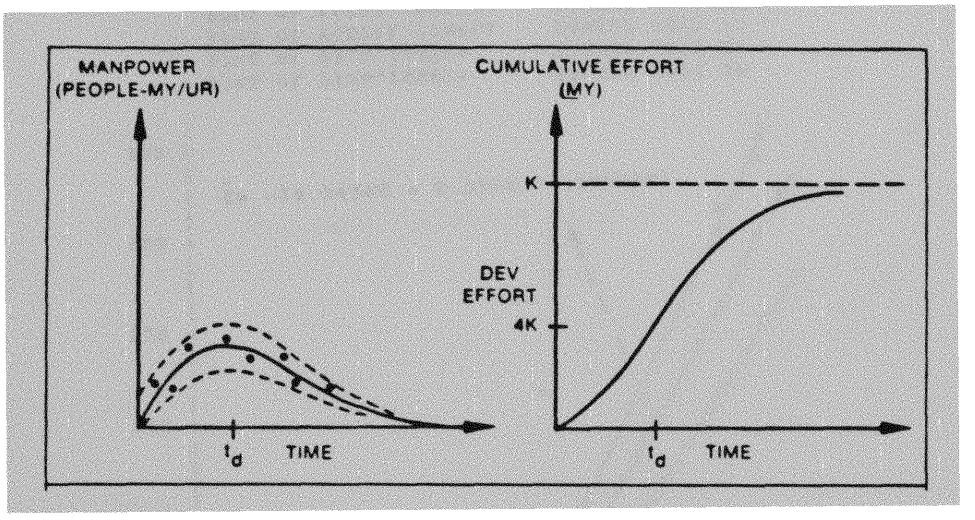


Figure 3. Manpower Pattern of Software Development for Large Software Systems Interpreted by Putnam

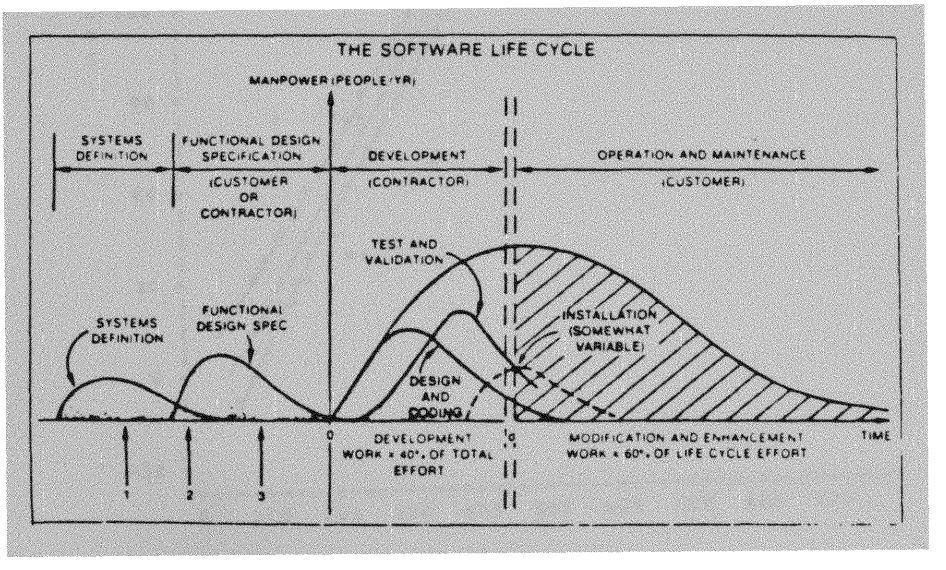


Figure 4. Software Life Cycle with Stages of Software Development Interpreted by Putnam

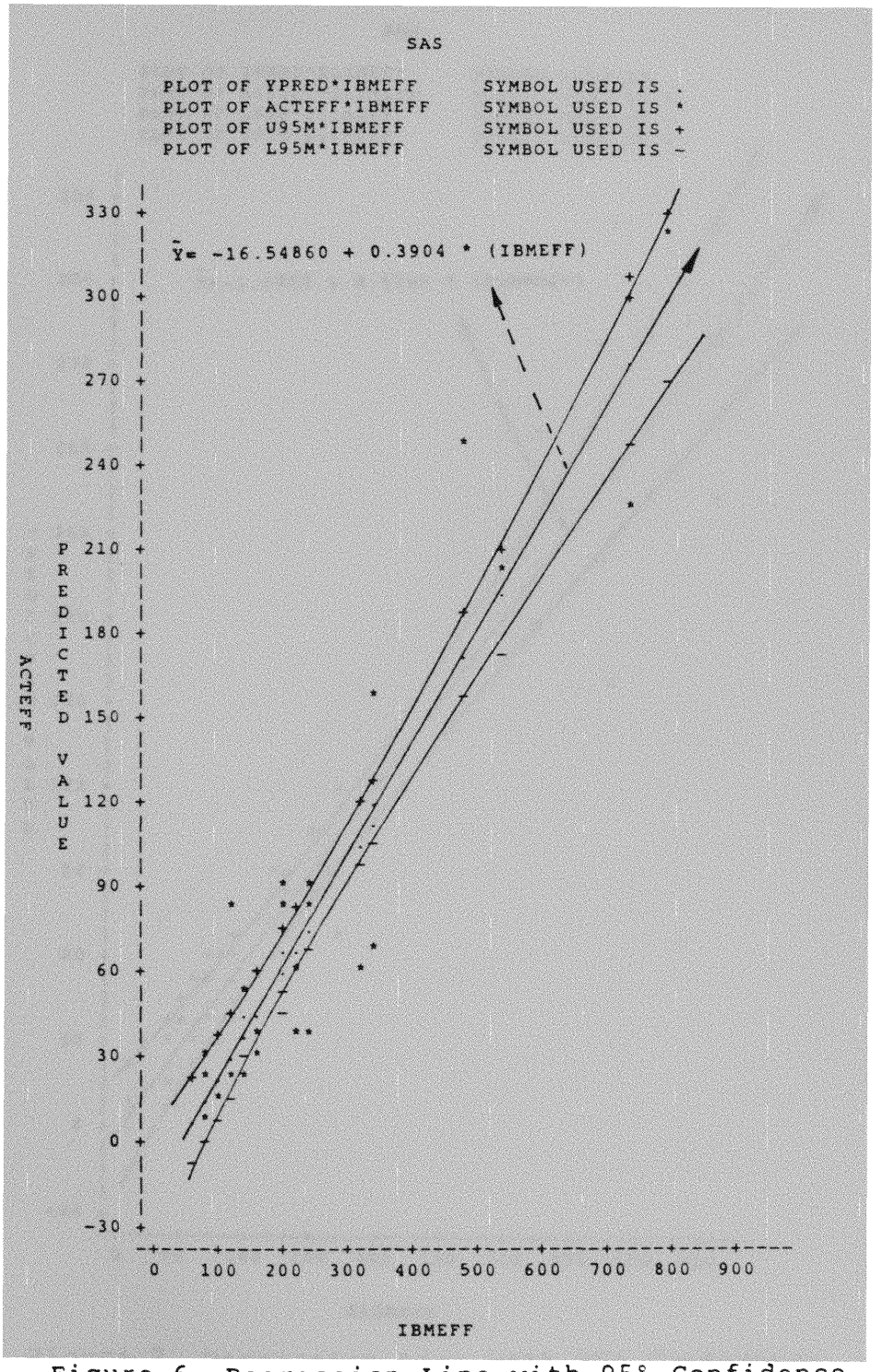


Figure 6. Regression Line with 95% Confidence Interval for ACTEFF on IBMEFF

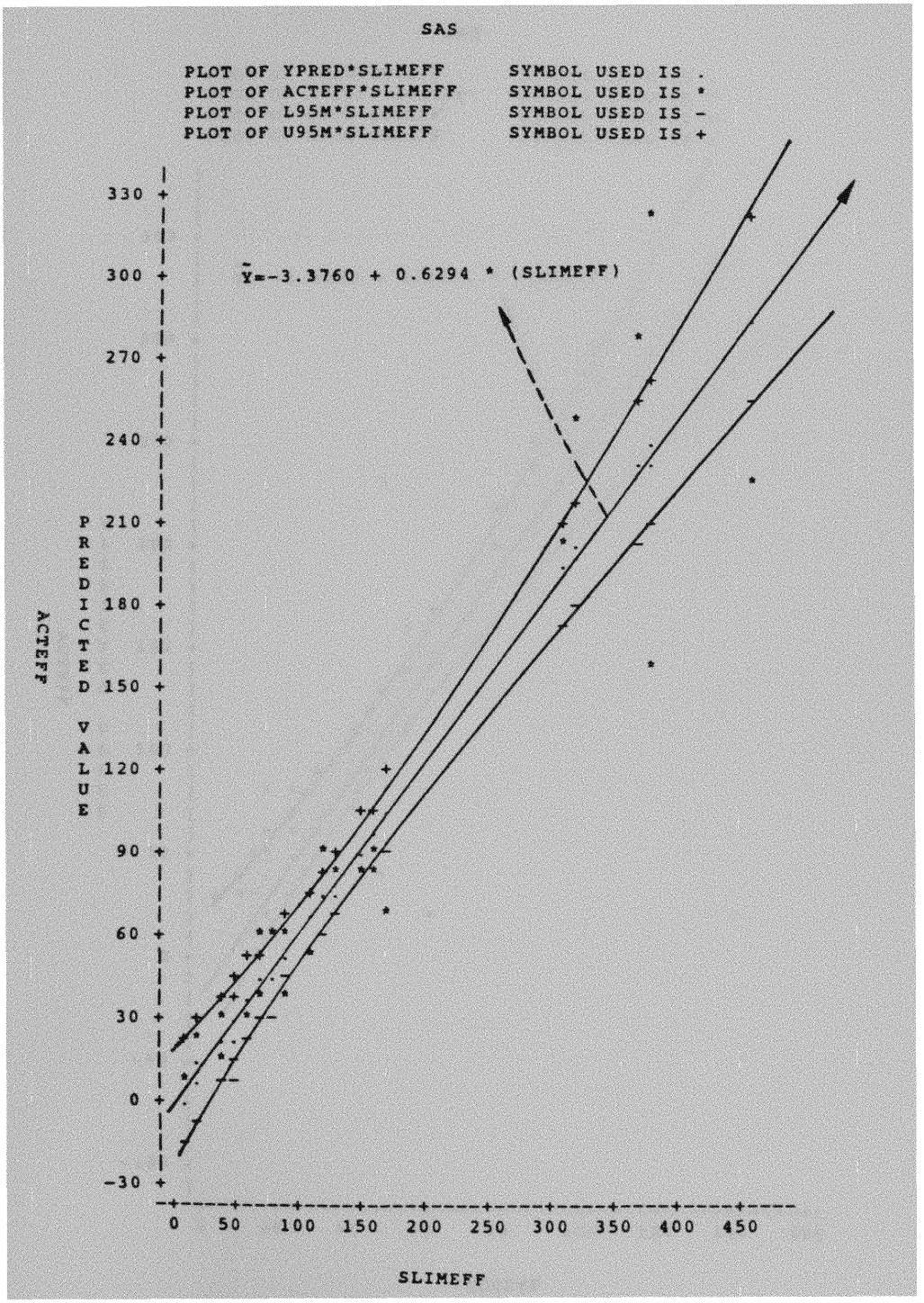


Figure 7. Regression Line with 95% Confidence Interval for ACTEFF on SLIMEFF

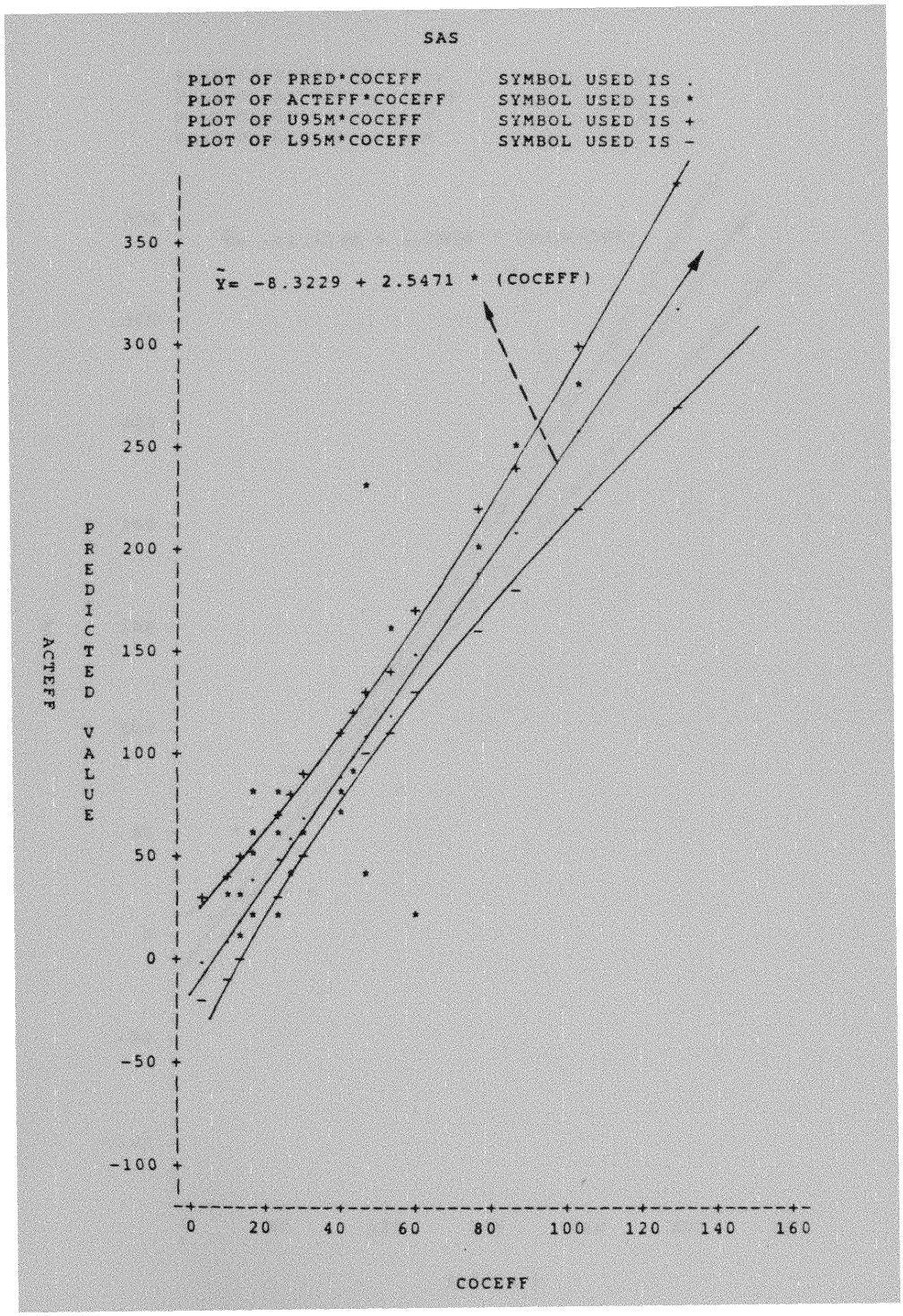


Figure 8. Regression Line with 95% Confidence Interval for ACTEFF on COCOMOEFF

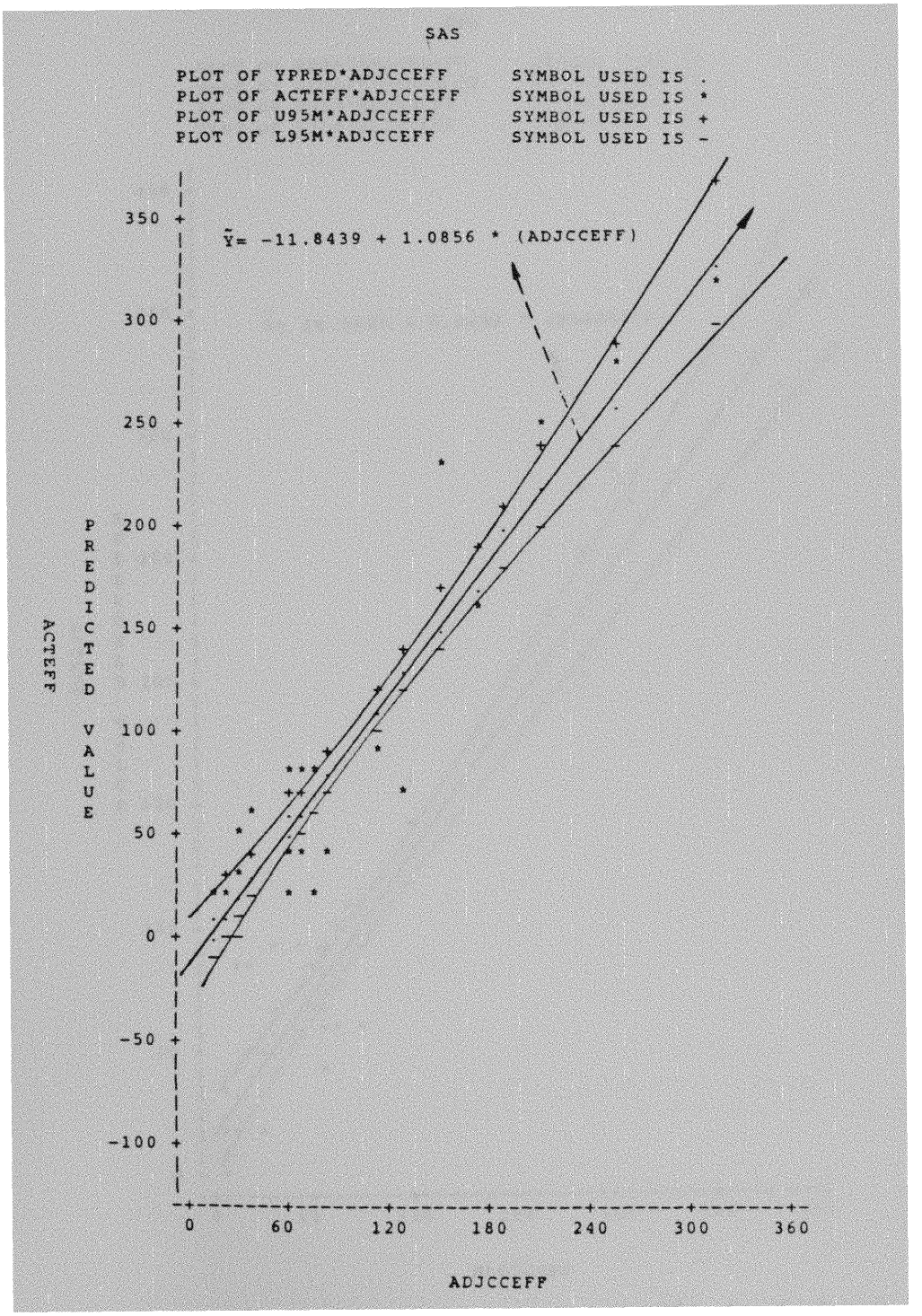


Figure 9. Regression Line with 95% Confidence Interval for ACTEFF on ADJUSTEFF

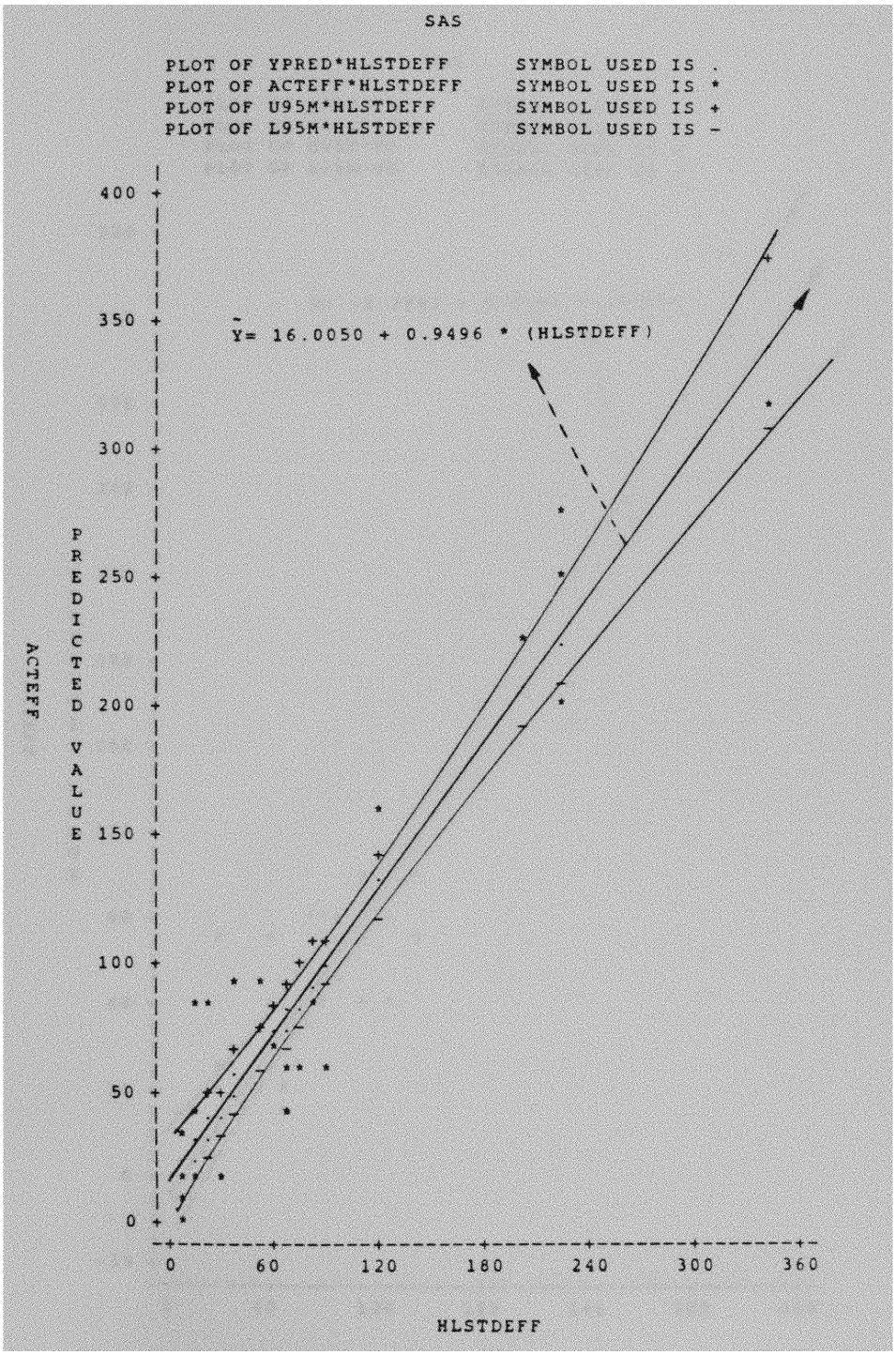


Figure 10. Regression Line with 95% Confidence Interval for ACTEFF on HLSTDEFF

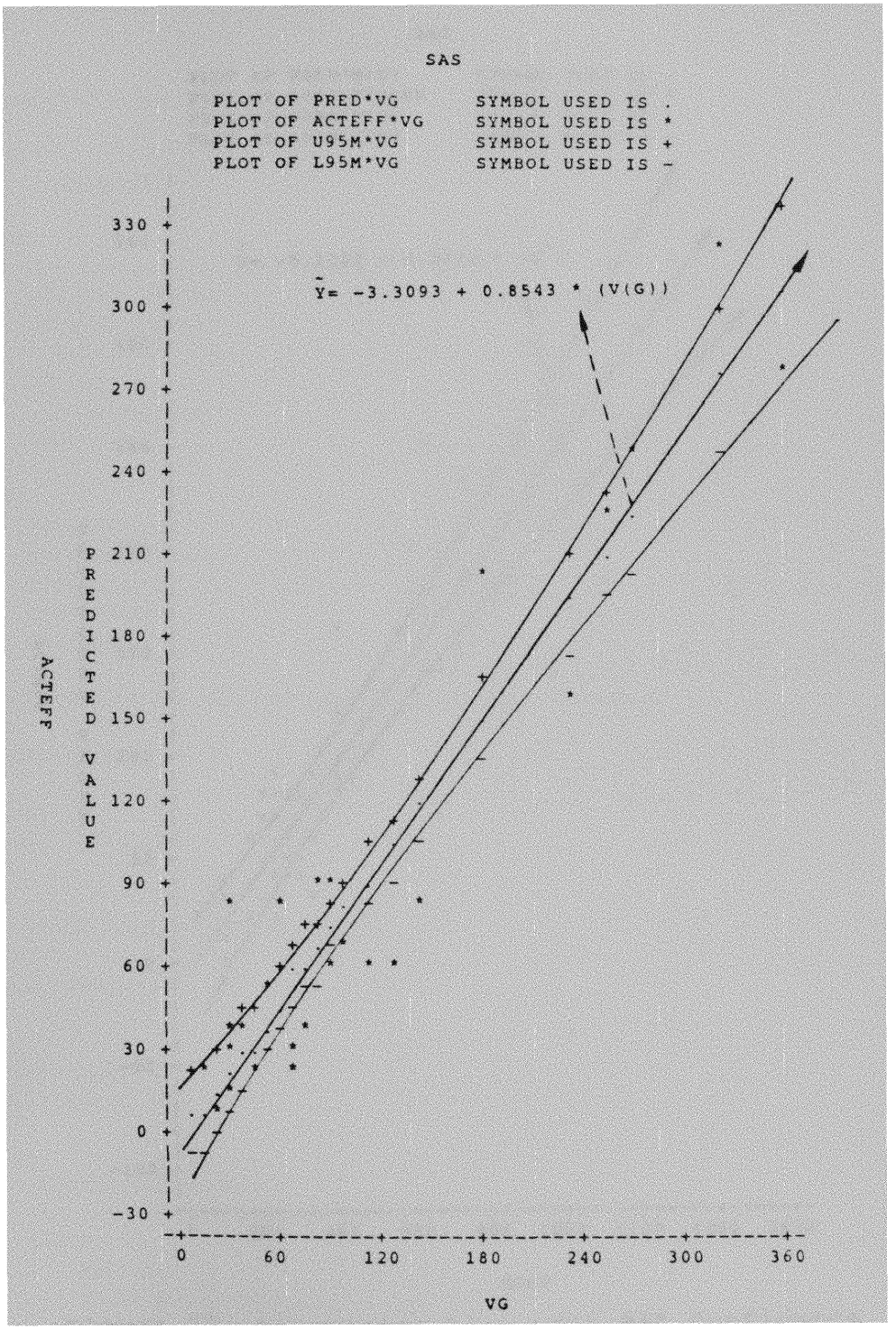


Figure 11. Regression Line with 95% Confidence Interval for ACTEFF on V(G)

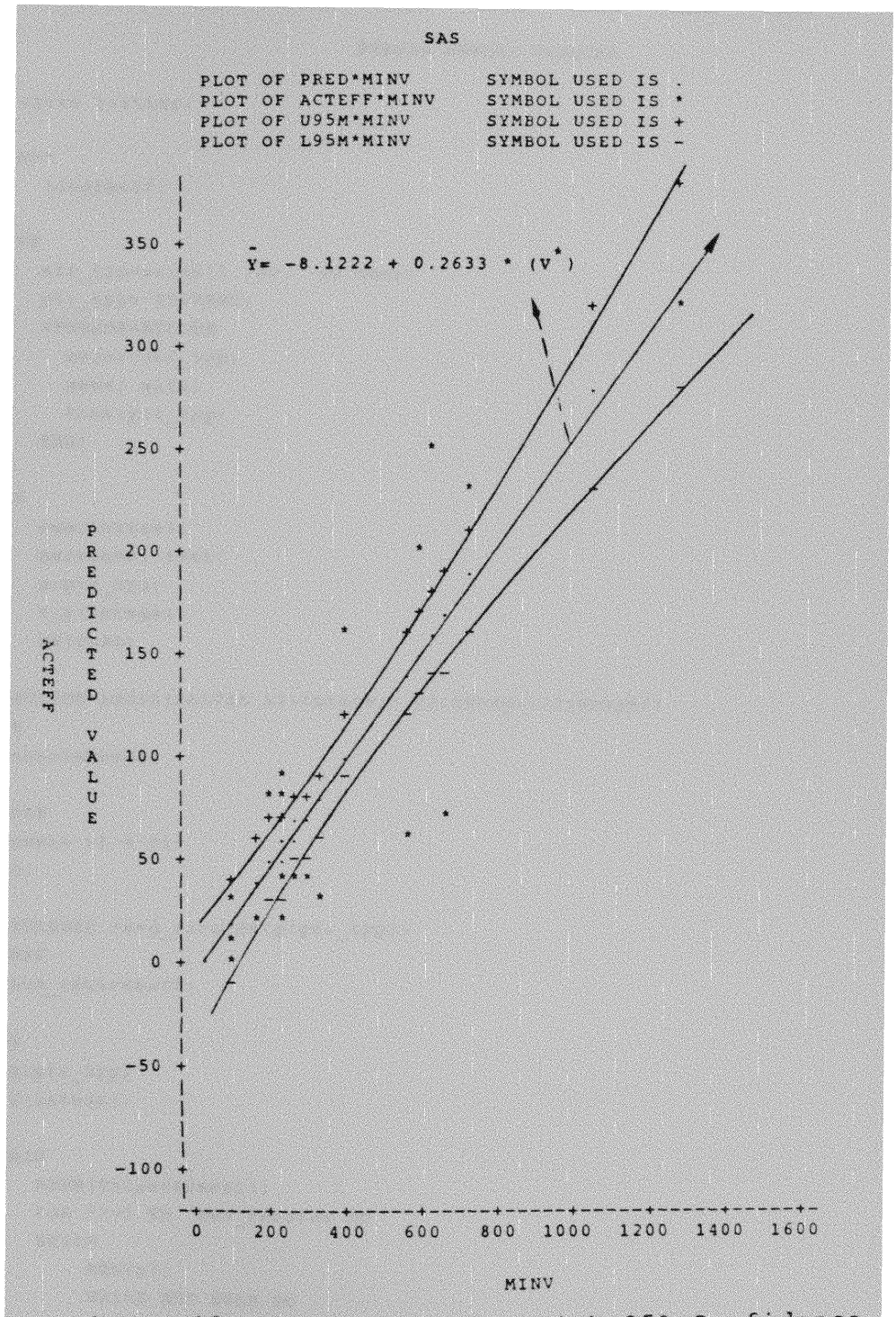


Figure 12. Regression Line with 95% Confidence Interval for ACTEFF on V*

APPENDIX A

Sample PASCAL Program

```
program testing;

CONST
    length=16;

TYPE
    str_typ=array[1..30]  of char;
    ptr_typ=^STUDENT;
    STUDENT=RECORD
      major:str_typ;
      name: alfa;
      link:ptr_typ;
    END;

VAR
    sum:integer;
    personnel:text;
    p:ptr_typ;
    x,y:integer;
    ch:char;

FUNCTION addition(VAR x1:integer; y1:integer):integer;
VAR
  sum:integer;

BEGIN
  sum:= x1 + y1;
END;

PROCEDURE read_records(p:ptr_typ);
CONST
  num_records=10;

VAR
  q:ptr_typ;
  I:integer;

BEGIN
    REWRITE(personnel);
    FOR I:=1 TO  num_records DO
    BEGIN
        NEW(p);
        WHILE NOT EOLN DO
        BEGIN
          read(p^.name);
          p^.major:= 'Computer Science              ';
          p^.link:=NIL;
        END;(* while*)
        q:=p;
```

```
        write(personnel,p);

    END;(* for *)
END;(* proc *)

(* main *)
BEGIN
    writeln(" for test purpose ");
    sum:= addition(x,y);
    read_records(p);
END.
```

APPENDIX B

Output of the Metrics Analyzer Program

```
Output of Block Name : ADDITION  ( Level : 1 )
  <*   Local variables  *>
  SUM           Counts :  2
  Y1            Counts :  2
  X1            Counts :  2

  <*  Deliminator Counts  *>
  :=            :     1
  +             :     1
  :             :     4
  ;             :     5
  (             :     1
  {             :     0
  (*            :     0
  '             :     0
  "             :     0

<*  Reserved Words Counts  *>
BEGIN        :     1
END          :     1
VAR          :     2
INTEGER      :     4

<*  Constant table is empty  *>

<*  Literal table is empty  *>

<* Global references in block ADDITION *>

Unique global symbols :     0
Total global symbols  :     0
Maximum distance      :     0
Average distance      :     0

Parameters : 2  Variable : 1  Value : 1  Func/Proc : 0

Unique Operand :     4
Unique Operator:     7
Total Operand  :    10
Total Operator :    15

Output of Block Name : READ RECORDS  ( Level : 1 )
<*   Local variables  *>
I             Counts :  2
Q             Counts :  2
LINK          Counts :  0
NAME          Counts :  0
```

```
MAJOR          Counts :  0
NUM_RECORDS    Counts :  2
P              Counts :  7
LINK           Counts :  1
NAME           Counts :  1
MAJOR          Counts :  1


(*  Deliminator Counts  *)
  :=           :      4
  =            :      1
  :            :      3
  ;            :     14
  ,            :      1
  .            :      3
  (            :      5
  ^            :      3
  [            :      0
  (*           :      2
  '            :      1
  "            :      0

(*  Reserved Words Counts  *)
BEGIN        :      3
CONST        :      1
DO           :      2
END          :      3
FOR          :      1
NIL          :      1
NOT          :      1
TO           :      1
VAR          :      1
WHILE        :      1
INTEGER      :      1
EOLN         :      1
NEW          :      1
READ         :      1
REWRITE      :      1
WRITE        :      1

(*  Constant Table   *)
1               1
10              1

(*  Literal Table  *)
COMPUTER SCI   1

(* Global references in block READ_RECORDS  *)

PERSONNEL      lvl:  1   cnt:   2
PTR_TYP        lvl:  1   cnt:   2
Unique global symbols :      2
Total global symbols  :      4
```

```
Maximum distance       :      1
Average distance       :      1.00

Parameters : 1  Variable : 0  Value : 1  Func/Proc : 0

Unique Operand :     17
Unique Operator:     18
Total Operand  :     26
Total Operator :     51

Output of Block Name : TESTING  Level : 0 )
(*   Local variables  *)
READ_RECORDS Counts :  2
ADDITION     Counts :  2
CH           Counts :  1
Y            Counts :  2
X            Counts :  2
P            Counts :  2
LINK         Counts :  0
NAME         Counts :  0
MAJOR        Counts :  0
PERSONNEL    Counts :  3
SUM          Counts :  2
STUDENT      Counts :  2
LINK         Counts :  1
NAME         Counts :  1
MAJOR        Counts :  1
PTR_TYP      Counts :  5
LINK         Counts :  0
NAME         Counts :  0
MAJOR        Counts :  0
STR_TYP      Counts :  2
LENGTH       Counts :  1

(*  Deliminator Counts  *)
  :=           :      1
  =            :      4
  :            :      8
  ;            :     16
  ,            :      2
  .            :      1
  (            :      3
  [            :      1
  ^            :      1
  ..           :      1
  {            :      0
  (*           :      2
  '            :      0
  "            :      1

(*  Reserved Words Counts  *)
ARRAY        :      1
BEGIN        :      1
```

```
CONST        :      1
END          :      2
FUNCTION     :      1
OF           :      1
PROCEDURE    :      1
RECORD       :      1
TYPE         :      1
VAR          :      1
INTEGER      :      2
CHAR         :      2
TEXT         :      1
WRITELN      :      1
ALFA         :      1

(* CALL is counted as a unique operator *)

(*  Constant Table   *)
30              1
1               1
16              1

(*  Literal Table  *)
 FOR TEST PU   1

(* Global references in block TESTING  *)

Unique global symbols :      0
Total global symbols  :      0
Maximum distance      :      0
Average distance      :         0

Parameters : 0  Variable : 0   Value : 0  Func/Proc : 0

Unique Operand :     26
Unique Operator:     19
Total Operand  :     42
Total Operator :     50

((*  Total Counts  *))

(*  Deliminator Counts  *)
  :=            :      6
  +             :      1
  =             :      5
  :             :     15
  ;             :     35
  ,             :      3
  .             :      4
  (             :      9
  [             :      1
  ^             :      4
  ..            :      1
  {             :      0
```

```
   (*           :      4
   '            :      1
   "            :      1

<*  Reserved Words Counts  *>
ARRAY        :      1
BEGIN        :      5
CONST        :      2
DO           :      2
END          :      6
FOR          :      1
FUNCTION     :      1
NIL          :      1
NOT          :      1
OF           :      1
PROCEDURE    :      1
PROGRAM      :      1
RECORD       :      1
TO           :      1
TYPE         :      1
VAR          :      4
WHILE        :      1
INTEGER      :      7
CHAR         :      2
TEXT         :      1
EOLN         :      1
NEW          :      1
READ         :      1
REWRITE      :      1
WRITE        :      1
WRITELN      :      1
ALFA         :      1

<* CALL is counted as a unique operator *>

<*  Constant Table   *>
16              1
30              1
10              1
1               2

<*  Literal Table  *>
 FOR TEST PU   1
COMPUTER SCI   1

-------------------------------------------------
Total operators              :        117
Total operands               :         79
Total distinct identifiers   :         45
Total distinct operators     :         26
Vocabulary size              :         71
Program length               :        196
Expected program length      :        369.3448
```

Program Volume	:	1205.3504
Potential volume	:	52.8148
Program level	:	0.0438
Intelligence	:	52.8148
Effort	:	27508.7754
Implementation time	:	1528.2653

REFERENCES

1. Albrecht, Allan J. and Gaffney John E. Jr., "Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation," IEEE Transactions on Software Engineering, Vol. SE-9, No.6, November 1983.

2. Boehm, Barry W., "Software and its Impact: A Quantitative Assessment," Datamation, pp.48-59, May 1973.

3. Boehm, Barry W., "Software Engineering Ecomonics," Prentice-Hall, Inc., Englewood Cliffs, N.J., 1981.

4. Boehm, Barry W., "Software Engineering Economics," IEEE Trans. Software Eng., vol. SE-10, No.1, pp.4-19, January 1984.

5. "BMDP Biomedical Computer Programs P-Series", University of California Press, 1977.

6. Demarco, Tom, "Controlling Software Projects Management, Measurement & Estimation," Yourdon Press, New York, 1982.

7. Farely, Richard E., "Software Engineering Concepts," McGraw- Hill, New York, 1985.

8. Freiman, Frank R. and Park Robert E., "PRICE Software Model-Version 3: An Overiew," Proceedings, IEEE-PINY Workshop on Quantitative Software Models, Catalog No. TH0067-9, pp.32-39, October 1979.

9. Halstead, Maurice H. "Elements of Software Science," Elsevier North-Holland, Inc., New York, 1977.

10. Henry, F. Dircks, " Gruman's Software Cost Estimating Model," The proceedings for the National Aerospace and Electronics Conference, pp.674-683, May 19-21, 1981.

11. Henry S., and Kafura D., "Software Structure Metrics Based on Information Flow," IEEE Trans. on Software Engineering, Vol. SE-7, No.5, pp.510-518, Sept. 1981.

12. Jensen, Randall W., "An Improved Macrolevel Software Development Resource Estimation Model," Proceedings 5th ISPA Conference, pp.88-92, April 1983.

13. Jensen, Randall W., "Sensitivity Analysis of The Jensen Software Model," Proceedings 5th ISPA Conference, pp.384-389, April 1983.

14. McCabe, Thomas J. "A Complexity Measure," IEEE Transactions on Software Engineering, Vol. SE-2, No.4, December 1976.

15. Meyer, Stuart L., "Data analysis for Scientists and engineers," Jonh Wiley & Sons, Inc., New York, 1975.

16. Montgomery, Douglas C., and Elizabeth A. Peck, "Introduction to Linear Regression Analysis," John Wiley & Sons, Inc., New York 1982.

17. Navlakha Jainendra K., and Qiang Li," Development of Software Interface Metrics", Phase I Report submitted to NCR, April 1985.

18. Ott, Lyman, "An Introduction to Statistical Methods and Data Analysis," PWS Publishers, Boston, Massachusetts 1984.

19. Putnam, Lawrence H., "A General Empirical Solution to Macro Software Sizing and Estimating Problem," IEEE Trans. on Software Eng., pp.345-361, July 1978.

20. Putnam,Lawrence. H., "Example of an Early Sizing, Cost, and Schedule Estimate for an Application Software System," Proceedings COMPSAC, 1978.

21. Putnam, Lawrence. H. and Fitzsimmons, A., "Estimating Software Costs, "Datamation, pp.189-198, Sept. 1979; Continued in Datamation pp.171-178, Oct. 1979 and pp.137-140 Nov. 1979.

22. Putnam, Lawrence H., "Tutorial Softare Cost Estimating and Life-Cycle Control," IEEE Catalog NO. Eho 165-1, Library of Congress no.80-83083, 1980.

23. "PRICE Software Model: Supplemental Information," RCA, Cherry Hill, N.J. March 1978.

24. "SAS User's Guide: Statistics", SAS Institute Inc., North Carolina, 1985.

25. Shapiro, Samuel S. & Gerald J. Hahn, "Statistical Models in Engineering," John Wiley & SOns, Inc., New York 1968.

26. Walston C.E. and Felix C.P., "A Method of Programming Measurement and Estimation," IBM Systems J., pp.54-73 June 1977.

27. Wang Institute, "WICOMO User's Manual," June 1984.

28. Wolverton, R.W., "The Cost of Developing Large Scale Software," TRW Inc., IEEE Tran. on Computer, VOL. C-23, No. 6, June 1974.

29. Yin B.H. and Winchester J.W., "The Establishment and Use of Measures to Evaluate the Quality of Software Designs, " Proceedings of the software and assurance workshop, pp.45-52, Nov. 1978.