Development of a Resource Scheduling Model

by

Mark G. Gintner

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Master of Computer Science and Applications

in

Computer Science

APPROVED:

Sallie M. Henry, Chairman

Dennis G. Kafura

James D. Arthur

June, 1986

Blacksburg, Virginia

Development of a Resource Scheduling Model

by

Mark G. Gintner

Sallie M. Henry, Chairman

Computer Science

(ABSTRACT)

The purpose of this study is to find models that best represent the software development process of IBM Federal Systems Division (FSD). The primary area of research is to evaluate existing effort estimating models and to develop a model best suited to IBM FSD's environment. The method of evaluation is an analysis of three existing models and their performance on a database of project information. The models used in this study included the Basic COCMO model by Boehm, the COPMO model by Thebaut and Putnam's macroestimating model. After extensive evaluation a hybrid effort estimating model was developed, and proven to be the best predictor of effort.

Acknowledgements

I would like to thank my advisor, Sallie Henry for her support, guidance and patience throughout this project, and my education in computer science. I also want to express special thanks to Dennis Kafura and James Arthur for their help and willingness to contribute extra time in Sallie Henry's absence. I am grateful to my committee for not only the opportunity to work with them, but also for their willingness to allow me to have a role in the direction of this project.

The research which is the source of this thesis has been sponsored by International Business Machines Corporation (Federal Systems Division, Manassas, VA). I am very grateful to IBM for their support. It is support such as theirs that will lead to a better tomorrow for computer science. I want to deeply thank Robert Cruickshank for his continual willingness to help, and strong support throughout the entire process.

Much of the success of my research is due to friends who were willing to help whenever I needed them. My special thanks go to Janet Davis, Steve Wake, Calvin Selig, Marc Slack and Therese Martin. Each of these people has played a special role and helped me to keep my sanity throughout a difficult process.

Last and not least, I would like to thank my parents and family for their support and help throughout my entire education.

Table of Contents

List of Tables

Chapter 1

Introduction

A large portion of the money spent in the computer industry is spent on software. By 1979, 56% of the total computer dollar was spent on software, 80% of the money spent by the Defense Department on computers was spent on software [Sho83]. The percentage of money spent on software is increasing due to the decreasing cost of hardware and the increasing cost of personnel and management in the software development area. There are four types of software producers in the computer industry: software contractors (to the Defense department, NASA, and other branches of the governement), hardware vendors (developing software for their own systems), custom software houses (for private industry and the home computer user) and the companies which internally develop their own software (payroll, marketing, accounting, and many other applications).

The need to control cost is a common concern of all four types of software producers. The first three types of developers are in highly competitive market. The internal developer is generally underfunded to handle only a portion of the requested workload. All four must, in some form, estimate the cost of development before the project begins. The need for accuracy in cost estimation is very important, especially for the government contractor, since overestimation will lose the contract and underestimation will cause a loss of profit.

Therefore, the software development industry must possess a means of estimating cost as close to the actual cost as possible. The method which has emerged in the computer industry to perform this task is predictive cost models. Predictive cost models are equations which generate cost, effort or duration information from any number of project parameters. Effort represents the number of man-months (MM) of work required to develop a project. Duration is the amount of time required for a given staff to develop a project. Effort and duration can be easily converted to cost by multiplying these factors by a company's cost factor for a man-month or calendar month respectively. For example

$$Cost = C \times MM$$

where C is the company's cost factor. This cost factor is determined by observing and measuring the development process over time. The parameters which are input to a model can include project size, complexity, staff makeup, environment and many more.

The initial attempts at modelling were mostly carry-overs from other technical areas. These usually did not work since projects often experienced overruns approaching two hundred percent and a small portion of the projects were never completed [DeM82]. Early estimates were made by using some formal estimation technique to derive an initial value, and then doubling that value to achieve a fairly accurate estimate.

In the early 1960's, attempts were made to model software development. Initially, a method to determine programmer producivity rates (lines of code per month) was used to calculate duration based on project size. Other researchers of the time tried to find the factors which affected the development of software [Moh81]. In general the early modelling period produced good background information but no really valuable models.

However, during the period from 1970 to 1980 there were a large number of efforts from both academia and industry, to develop "the predictive cost model". Out of the flourish of activity came at least ten slightly different solutions to essentially the same problem. Most of the new models were developed by modelling the project behavior of some historical database of collected project information (empirically formed models). Typically the model would do well for the database and for the organization from which the data was collected, but rarely perform well on other collected sets of data. This limiting factor essentially caused the creation of "custom-tailored" models.

The most recent efforts in this area have gone in two directions. The first has been to develop models which have a broader spectrum of use. This goal has been attempted by developing models over several of the collected historical databases. People have also tried to broaden functionality of existing models which were successful in certain applications. The other direction of current work becoming prevalent is the commercialization of the modelling concept. Previously most of the use of predictive models was limited to large development efforts, whereas estimating packages are becoming available on micros as well as in the software systems provided by major vendors [Rub83]. Due to the broad availability, development efforts of all sizes can make use of estimation techniques. This growth can only be advantageous to the field, due to the increased number of users who will generate a greater wealth of data for future work.

The abundance of models that have been developed, although each is slightly unique, can easily be divided into three classes. This classification is similar to the classification used by Basilli [Bas80]. The three classes are: single independent variable (SIV) models, multiple independent variable (MIV) models and theoretical models.

SIV Models

The SIV class of models consist of models based on size or source lines of code (SLOC). All SIV models use SLOC as the independent variable, and are empirically formed. The class can be subdivided into two areas. This first area are those models which use just the total SLOC (excluding inline documentation and undelivered software such as test drivers), or the total number of machine instructions. Examples of this type of model are as follows.

The Walston and Felix model based on IBM data is probably the best known of the SIV models. The primary result which they found for their data was was that:

$$Effort = 5.2 \times KSLOC^{0.91}$$

where KSLOC equals one thousand SLOC [Wal77]. A similar effort was undertaken at the Software Engineering Lab at the University of Maryland. Their results were:

$$Effort = 1.4 \times KSLOC^{0.93}$$

Both of these studies found relationships between SLOC and documentation pages, SLOC and duration, effort and duration, and effort and staff size. Although the models were calculated for different databases of information, the equation constants were within one standard error of each other [Bas78].

The General Research Corporation (GRC) model is in the same form as the above models but produces a much higher estimate. Their model is:

$$Cost = 0.232 \times SLOC^{1.43}$$

where Cost would be divided by a cost/MM value to determine effort. The GRC model also includes an optional multiplier which has a value of less than one. The multiplier is used to reduce the cost estimate, if the development environment has resources which exceed the needs to develop the project [Moh81].

The Basic Constructive Cost Model (COCOMO) by Boehm is a well known model. His model has a effort equation and duration equation for each of three classes of software. The equations for each of the classes and further discussion on the model are in the next chapter [Boe81].

The model by Schneider is slightly different from the typical SIV model. This model is derived from the equations of Halstead's software science theory [Hal77]. The Halstead equations include variables for the language level and language vocabulary size as well as program length. Schneider calculated an average for language level and language vocabulary size leaving size as the remaining independent variable [Moh81]. The resulting model is:

$$Effort = 0.3 \times KSLOC^{1.83}$$

The second type of SIV models are those models which divide the SLOC into specific types of code. The SLOC for each type is then multiplied by a constant for that type to calculate effort. The individual effort values are then added up to give effort required for the overall project. The Wolverton model is the best known of these models. Wolverton uses the code categories: control, input/output, pre/post-processor, algorithm, data management, and time critical. Additionally, for each code type he has an easy, medium and hard constant to represent the difficulty of that type of code. This technique can be used to create a range or to accommodate varying difficulty within a project [Put77].

In the Kustanowitz model [Moh81] the same code categories are used. However, the model uses an upper and lower productivity constant rather than complexity multipliers to get effort. After adding up the individual effort ranges a final best/worst range is created.

The Aerospace model [Moh81] uses slightly different code categories which are: real-time, nonreal-time system software and nonreal-time operational software (interactive, on-line, mathematical and string manipulation). It also includes variables for system complexity and development environment.

The Aron model is one of the earlier models and uses the programmer productivity rate philosophy. The code is divided into modules by code type. Module difficulty (easy, medium, difficult) is determined by the number of interactions the module has with other modules. The difficulty and expected duration are then used to determine the productivity rate from a table of values. The estimated size is divided by the rate to give the needed effort. This process is done for all modules giving an overall project effort estimate [Sho79].

The Doty model used a subset of the code categories: command and control, scientific, business, and utility. It also included a category for all types. The model supplied a set of equations for high level code and another set for assembler. Doty's model is unique in that it used equations of the form:

$$Effort = a \times KSLOC^{b}$$

whereas most SIV models of this type use equations of the form:

$$Effort = a \times KSLOC$$

This type of model allows for curves with either decreasing or increasing slopes. The command and control curve has an increasing slope, and the business and utility equations have decreasing slopes. These equations then infer that as command and control modules get larger they require increasingly more effort and that as business and utility modules increase in size the additional needed effort slightly decreases [Sho79].

MIV Models

The MIV class of models are those models which use two or more independent variables in an empirically formed model. This class also has two semi-distinct approaches. The first approach is best called a subjective method. Models of this approach use objective values for subjective parameters such as programmer experience, complexity, and development environment reliability. The other approach is an objective method. This method uses only objective values such as size, average staff size and language to be used.

The subjective MIV models which appear in literature are the Price-S system and the Intermediate COCOMO model. The Price-S system was developed by RCA in the mid 70's as a byproduct of their cost estimating system for military hardware. The inputs for Price-S system are: SLOC, type of code, level of new/old code, experience and ability of staff, project difficulty (impactable factors), specifications and requirements and hardware resources. These factors are then fed into an undisclosed parametric equation. The Price-S system has two other modes of operation. ECIRP mode allows the model to run in reverse based on groups past projects. This option allows a manager to get a better understanding of the development environment. The Design-To-Cost mode allows estimators to see the impacts of trying to meet specified funding levels.

Price-S is most likely the most commercially successful of all the models. It is the sole quantitative model used by the Avionics Division of Honeywell Inc. [Bjo79]. Price-S is used together with the Putnam model by General Electric's Space Division [Jun79]. The Price-S model is the primary model of the Department of Air Force Space and Missile Systems Organization estimating methodology, which also includes the Doty model, the Walston and Felix model and the Putnam model [Las79]. The above three groups have reported good initial results with the Price-S system but have supplied no quantitative results.

The Intermediate COCOMO model by Boehm uses the base equations of Basic COCOMO for the initial effort estimate. The user must then determine a series of cost drivers in four areas. The four areas are product attributes, hardware attributes, personnel attributes and development environment attributes. Each driver is essentially a multiplier which can increase, decrease or not affect the estimate. The overall package of Boehm models has been given some critical acclaim, but it has also been closely studied and yielded some doubts in accuracy and practical application [Bry83].

The System Development Corporation (SDC) completed one of the pioneer studies in the modelling area in early 60's for the U.S. Air Force. Their work looked for the factors which affected productivity. They initially found ninety-four possible factors; of which only eleven of them were statistically relevant. The factors were then formulated into a model which included a multiplier for each factor giving effort as the result [Moh81].

Two other models of the same form as the SDC model are the Farr and Zagorski model, and the Naval Air Development model. Both models use thirteen productivity factors in a model with a multiplier for each factor. The models differ only in the multipliers for the factors [Moh81].

Objective MIV models are less common and more of a newer approach. The cost estimating model Halstead derives from his software science metric was the first model of this area. The inputs for the Halstead model include size (S), vocabulary size of the language used (V), and a language level value (L). The resulting equation is:

$$Effort = \frac{2.1433 \times 10^{-7} \times (S \times \log_2 V)^{1.5}}{L \times 5} \quad \text{[MOH81].}$$

Thebaut's Cooperative Programming Model (COPMO) is the other objective MIV model. Thebaut's model uses two components to estimate effort: programmer cost and the cost of coordinating the programmers. Programmer cost is determined by size and coordination cost is determined by the average staff size. The details of this model and its respective equations are discussed in the next chapter.

Theoretical Models

Theoretical models are the class of models which are not empirically derived. These models are based on a logical framework which explains project behavior. The logical framework used in both of the models of this type is the life cycle curve of development by Rayleigh [Nor77]. Putnam's model is the most well known of the theoretical models. Putnam's model atempts to quantify the effort needed at a given point in time, from design through maintenance. Full details of the Putnam model are presented in the next chapter.

Parr's model is another notable model of the theoretical class. The Parr model is also an implementation of Rayleigh's life cycle curve, with some minor differences. The primary difference is that the Parr model does not have effort start at zero due to the high level planning done before a project is initiated. The graph on the following page [Bas80] shows the difference between the results of the two equations. The Parr equation [Par80] is as follows:

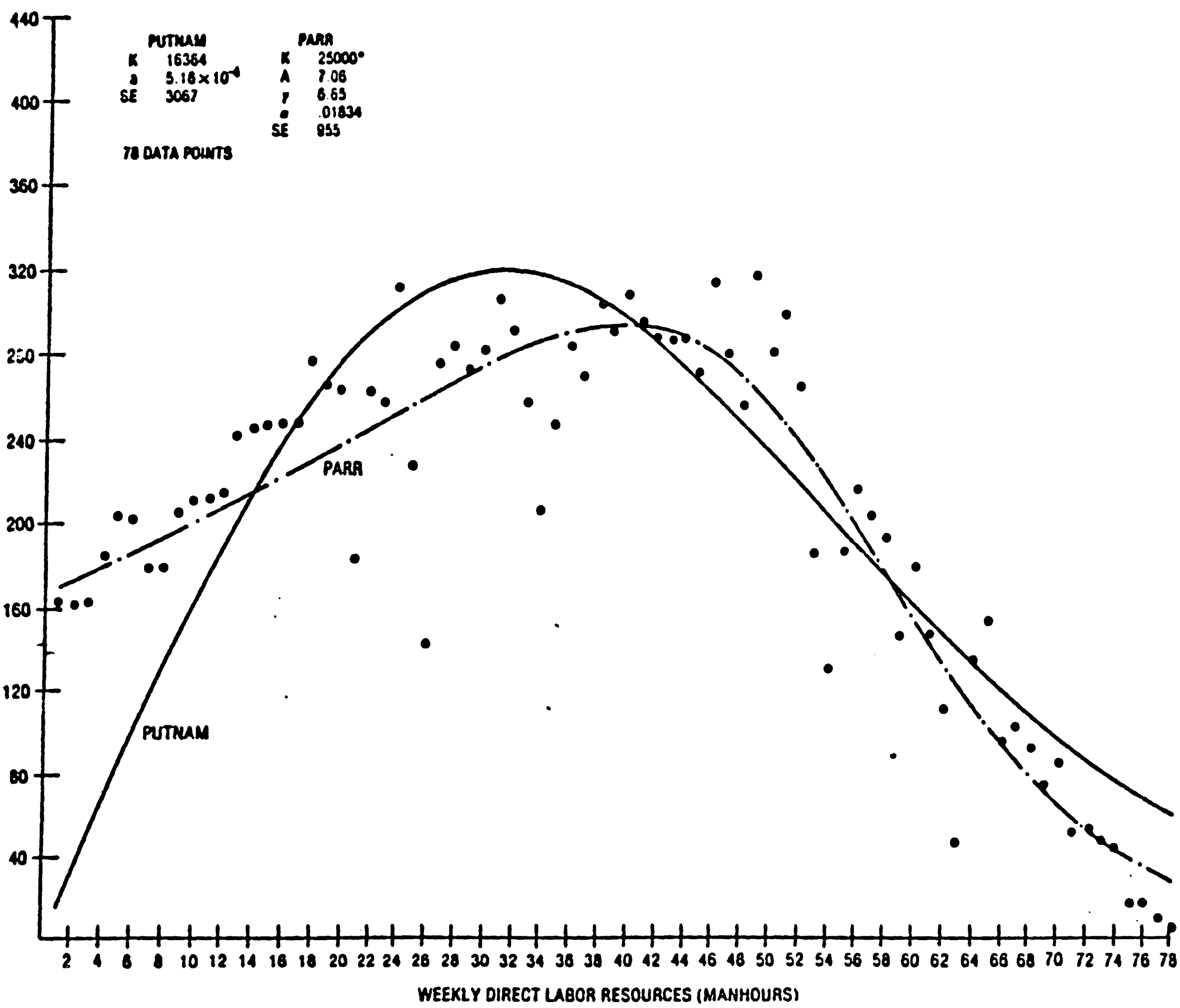
$$Y(t) = 1/4 \times sech^2((a \times t + c)/2)$$

where: Y(t) is the work rate at time t

a is the rate of increase of the work rate

t is the point in time.

In the following chapter, the general details of the data and the selection of models will be discussed. Chapter 3 includes a comprehensive discussion of the data, measurement techniques, and the results of various modelling approaches. Two techniques to enhance the models developed are discussed in chapters 4 and 5. Attempts to model duration are discussed in chapter 6. Chapter 7 is the conclusion and recommendations for further study.



The Parr Curve vs. the Putnam Curve

Chapter 2

Selection of The Models

This chapter describes, in detail, the motivations for the selection of several effort estimating models, and gives a detailed explanation of the models used in this study.

IBM Federal Systems Division (FSD) is the supporting organization for this study. Their goal is to find an effort estimating model which best predicts the behavior of their development environment. An effort to collect project data began approximately fifteen years ago. The data which was collected was based on a variety of defense contracting software projects, and was designed to contain a broad spectrum of project information. Given that each project is composed of one or more products, the database was meant to contain the type of product, language, estimated size, breakdown of code (new, modified and unchanged), duration, documentation pages, effort per phase of development, and computer-time cost for each product. To achieve accuracy in the database, the data must be collected during development rather than after the completion of the project. As exhibited by the sketchiness of some portions of the data, it is apparent that some of the data was added after development or that managers did not fully understand what information was needed for certain entries. Therefore, the database will be trimmed of projects which are poorly reported. Thirty-four projects composed of fifty-five products will remain in the database after eliminating some projects for the above reasons.

Since the project data was limited to size, effort, duration and some characteristics (language and an undisclosed product type), the selection of models to include in this study was limited. The subjective MIV models require information on complexity, project type or programmer experience. Due to this requirement, the subjective MIV models could not be selected. This data limitation extended to other classes of models and to the manner in which a selected model could be used.

In an effort to represent the three classes of models, a model from each was chosen. The selection was also affected by the IBM FSD's past use of the Putnam model. The Putnam model was used along with other models and produced mixed results.

Boehms's Basic COCOMO Model

Boehm's Basic COCOMO model was selected to represent the single variable class. The Basic COCOMO model consists of an effort predicting equation and a duration predicting equation and can be used on three types of software. The format of the effort predicting equation is:

$$Effort = a \times KSLOC^{b},$$

and the format of the duration predicting equation is:

$$Duration = c \times Effort^{d},$$

where the values a, b, c, and d are constants calculated with a best fit method.

Three types of software can be analyzed by this model. Organic mode is software developed by a team of experienced programmers who are familar with this particular type of software. The size of organic mode software is less than fifty KSLOC. This type of software is also characterized by basic adherence to requirements, and minimal use of complex or innovative architectures or algorithms. An example of an organic project is a simple payroll system.

Embedded mode is the other end of the spectrum and is characterized by a mixture of experience in programmers, strict adherence to requirements, as well as a complexity of external factors (hardware, regulations and procedures) which allow little flexibility in software. Embedded mode projects have a strict need to finish on schedule and these projects vary greatly in size. Embedded projects most often require innovative or complex approaches to solve the problems. An example of a embedded project is a large new operating system.

The third mode of software is referred to as semidetached mode, which is between the organic and embedded modes. The characteristics of semidetached mode are a primarily experienced team on a slightly complex project with a fairly strict need to adhere to requirements. A semidetached project could also be one which is composed of some organic traits and some embedded traits. An example of a semidetached project is a simple command control system.

The constants a, b, c, and d in the two predicting equations are calculated by some type best fit method to the historical data. The values which Boehm found for a variety collected data are as follows:

Organic: $Effort = 2.4 \times KSLOC^{1.05}$ $Duration = 2.5 \times Effort^{0.38}$

Semidetached: $Effort = 3.0 \times KSLOC^{1.12}$ $Duration = 2.5 \times Effort^{0.35}$

Embedded: $Effort = 3.6 \times KSLOC^{1.20}$ $Duration = 2.5 \times Effort^{0.32}$

The results generated by the above equations, with the values 10, 20, 40, 80 KSLOC are shown in Table 1. Table 1 also supplies the value staff, which is effort divided by duration, and represents the approximate number of personnel needed for the project.

Table 1. Organic and Embedded results from the Basic COCOMO Model

	Organic			Embedded		
KSLOC	Effort	Duration	Staff	Effort	Duration	Staff
10	27	9	3	57	9	8
20	56	12	5	131	12	11
40	116	15	8	302	15	20
80	239	20	12	692	21	35

The results of this example demonstrates many facts about the Basic COCOMO model. Effort is related to KSLOC at an increasing linear rate. Implementation of this model indicates that an increasingly larger effort is needed as project size increases. The rate of effort increase is larger for embedded software than for organic software since embedded software is more complex. The results also show that duration is related to KSLOC at a decreasing linear rate. Therefore as the size increases less additional time is needed to produce the additional code. Also, duration is related to effort at an even greater decreasing rate. This relationship implies that a larger number of programmers can develop much more code in a small amount of additional time. This type of relationship can cause the overmanning of projects since adding a few individuals to a project will have little effect on the duration equation. The startup time for an individual as well as the impact on programmers already on a project is generally significant enough to cause a serious increase in duration [Bro75]. As a corollary to this problem, the results show that an embedded project can be developed in the same time as similar sized organic project by adding people. This seems unlikely due to the difficult characteristics of an embedded project.

Thebaut's COPMO Model

The multivariable empirical class model selected was the Cooperative Programming Model (COPMO) by Thebaut. Thebaut's approach includes two methods for predicting effort and duration. The overall emphasis of this approach is a two part model:

$$Effort = E_p + E_c$$

The first portion of his model $E_{p,}$ represents the productive effort (design, development, and testing) expended by the programmer. $E_{p,}$ is calculated as

$$E_p = a + b \times KSLOC$$

where a and b are constants calculated with a best fit method. The second portion of the model $E_{c,}$ represents the cost of coordinating multiple programmers efforts. E_c is calculated as

$$E_c = c \times \overline{P}^{d,}$$

where $\overline{P}$ is the average personnel, and c and d are constants calculated with a best fit method.

Thebaut combined E_p and E_c in two different ways to create two overall unique model formats. The first approach was the first order model. This approach exactly followed the above equations with the exception of the exponent d. Therefore the first order model equation is:

$$Effort = a + b \times KSLOC + c \times \overline{P}$$

This format of the model was meant to be representive of an extension of a single independent variable model. Included under the first order model is a single independent variable equation:

$$Effort = a + b \times KSLOC.$$

Thebaut also used the same independent variables to predict duration. The equations for these were:

$$Duration = e + f \times KSLOC$$

$$Duration = e + f \times KSLOC + g \times \overline{P}.$$

The constants e, f and g are calculated with a best-fit separately for each equation.

Thebaut's second approach is the Power model. This model was formulated as:

$$Effort = a \times KSLOC^{b} \times \overline{P}^{c}.$$

This model represents a non-linear approach using cooperative independent variables. The Power model also included a single independent variable effort equation and two duration predicting equations:

$$Effort = a \times KSLOC^{b}$$

$$Duration = c \times KSLOC^{d}$$

$$Duration = c \times KSLOC^{d} \times \overline{P}^{e}.$$

The constants a, b, c, d and e in the above equations are calculated with a best-fit separately for each equation [The83].

The format of the COPMO model used in this study was a cross between the first order model and the power model. The model used was a direct implementation of his equation

$$Effort = E_p + E_c$$

yielding the equation

$$Effort = a + b \times KSLOC + c \times \overline{P}^{d}$$

based on the previously supplied definitions of E_p and E_c. Using the model in this format made it into a simple polynomial equation where a, b, c and d were calculated with a best-fit method.

Putnam's Macroestimating Model

The model used from the theoretical class of models is the Putnam's Life cycle model. Putnam's model is based on the life cycle curves developed by Norden [Nor77]. The basic equation which represents the life cycle curve is:

$$Y(t) = K/D^2 te^{-t^2/2D^2}$$

where: Y(t) is the percentage of the total manpower to be used,

K is the total man months,

D is the overall duration of the project,

t is the point in time.

An equation which predicts overall effort using size and duration can be derived from the life cycle equation. The format of the effort predicting equation is:

$$Effort = c \times \frac{KSLOC^3}{Duration^4}$$

where c is calculated through a best fit method.

The underlying assumption of the Putnam approach is that the amount of effort required for a project can be represented by a bell-shaped curve. This philosophy implies that a project with constant staffing throughout has wasted effort initially and uses more effort over a longer period than is needed to complete the project [Put77].

Putnam's approach is intuitively correct from a software engineering point of view. The multi-phased nature of software development, in an ideal situation, has a project begin with a few individuals designing the system, as the project progresses more personnel are required for development, testing and documentation. At the peak of the process, final design, development, various testing techniques, documentation and debugging are occurring simultaneously. After all the code is written the development effort has a decreasing number of individuals involved in final testing, debugging and documentation. Most software development efforts have generally been less structured. Projects often begin understaffed, which leads to predicted time overruns. When management realizes the problem, personnel are added to the project only to further delay completion. Projects often spend as much time on the last 5% of the project as spent on the first 95%. This backward situation occurs when poor design and development cause an exceedingly large system debugging phase.

The practical value of the Putnam model is somewhat debatable. The model is being used by General Electric Space Division [Jun79] and the U.S. Air Force Space and Missile Systems Organization [Las79]. However, neither presented any results on the performance of the model. The Putnam model did do well in the research done by Thebaut [The83], but the model displayed mixed results when used by IBM FSD, the sponsoring organization of this study.

The three models, COCOMO, COPMO and Putnam, described in this chapter are implemented and applied to the IBM FSD database. The results of this analysis are displayed and discussed in the next chapter.

Chapter 3

Further Data Definition

To aid in the explanation of the results, this chapter will first give an in depth discussion of the data and notation used in discussing the data. As mentioned in the previous chapter, a project is composed of one or more products. Each product has the following information: the effort required broken down by life cycle stage, the language it is written in, the product type, the breakdown of the SLOC (new, changed, modified retained and unmodified), and the duration of the development effort for that product. There are some other types of data collected that are not used in our study.

The effort information represents the product life cycle from design through integration. The effort information is collected at the task level within a life cycle stage, thus explicitly displaying utilization of effort.

There are five different product types. Examples of these could be operating systems software or data compression software. This field was coded for privacy and therefore became only a method of grouping products.

The language field has three different possible types. These types are: HO (high order), AS (assembler) or SL (signal processing language). Exactly which high order languages and assemblers used has also been extracted for privacy.

The breakdown of SLOC is in KSLOCs for new code, changed code, modified retained code and unmodified code. New code represents code which requires design and development. Changed code represents code that was previously developed but requires modification to be used. Modified retained code is code which was previously written for another product but required examination and little or no modification to be used in the new product. Unchanged code represents up to an entire product which was used in another product and requires no modifications to fit into the new product or project [Cru82].

Equivalent Measures

Products are composed of unique combinations of types of code. To compare two products, it was necessary to provide some measure of SLOC's. IBM FSD already had a measure called Equivalent Source Lines Of Code (ESLOC, KESLOC is one thousand ESLOC) which handles the different code types [Cru85]. The formula for this is:

$$\text{KESLOC} = \text{New KSLOC} + \text{Changed KSLOC} + 0.31 * \text{Modified Retained KSLOC} + .04 * \text{Unmodified KSLOC.}$$

Additionally, comparing a line of high order code with a line of assembler code is also inequitable. IBM FSD also has a method for equivalencing two products of different languages. The measure is Words Of Memory (WOM, one thousand WOM is a KWOM) which is computed after KESLOCs are computed [Cru85]. The formula for this was statistically computed to be:

$$KWOM = 1.2 \times HO\,KESLOC + 1.2 \times SL\,KESLOC + AS\,KESLOC$$

Result Measurement Techniques

Since this study evaluates three cost estimating models, some methods were needed to measure the quality of the results. The measuring techniques chosen were similar to the techniques used by Thebaut to compare cost estimating models [The83]. The primary measure in predicting the accuracy of the models is the measure of error. Error is defined as the difference between the actual effort and the estimated effort (Actual - Estimated). The four measures which were used are: $\overline{Error}$, R^2, $\overline{RE}$, and $\overline{MRE}$. $\overline{Error}$ is the most simplistic of the measures but a good overall indicator of the accuracy of a model. Its formula is:

$$\overline{Error} = \frac{\sum_{i=1}^{n} |Actual_i - Estimate_i|}{n}$$

Where n is the number of data points.

The R^2 value is the Coffecient of Multiple Determination. This value is a measure of the linearity between the actual and estimated values. The value of this measure is relative since it does not reflect how closely the values correspond. However, it is measure of how well the prediction equation is following the data points [The83]. The formula for this equations is

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (Actual_i - f(Estimate_i))^2}{\sum_{i=1}^{n} (Actual_i - \overline{Actual})^2}$$

Where n is the number of data points.

$\overline{RE}$ (average Relative Error) is a means of determining whether a prediction equation primarily overestimates or underestimates. The formula for this equation is

$$\overline{RE} = 1/n \times \sum_{i=1}^{n} \frac{(Actual_i - Estimate_i)}{Actual_i}$$

Where n is the number of data points.

The model is overestimating when $\overline{RE}$ is negative and underestimating when $\overline{RE}$ is positive. The ideal value for $\overline{RE}$ is close to zero on either side [The83].

The $\overline{MRE}$ (Average Magnitude of Relative Error) is a measure of the percent error over the whole set of points. The equation is defined as

$$\overline{MRE} = 1/n \times \sum_{i=1}^{n} \frac{|Actual_i - Estimate_i|}{Actual_i}$$

Where n is the number of data points.

and gives the percentage distance the estimates are from the actual value. The value of this equation is in using it along with $\overline{Error}$. A given group of products could have a low $\overline{Error}$, due to small average product size and still have a large $\overline{MRE}$. An example would be a group of products which have an average actual effort of 24 MM, if the average estimate was 48 MM, the $\overline{Error}$ would be 24 but the $\overline{MRE}$ would be 100%. Large products can be aided by $\overline{MRE}$ since $\overline{Error}$ can appear large, but is relatively close to the actual effort as shown by a small $\overline{MRE}$.

Overall Model

The simplest approach used was a single model over all products, and a single model over all projects. This technique was done for all three models. A best-fit correlation was done which generated the constants for the estimating equations in Table 2. Table 3 gives empirical results on the performance of the equations from Table 2.

The results of Table 3 are obviously less than desireable. However, they are somewhat explainable. For fifty-five products KWOM ranged from 2.7 to 225 and effort ranged from 18 to 1300 man-months. For thirty-four projects KWOM ranged from 2.7 to 1067, and effort ranged from 18 to 7106 man-months. Neither group increased in a linear manner. Both sets of points produced a scatter of points with no trackable pattern. The nonlinearity implies that there is not one type of behavior for either products or projects. A possible explanation is that the points need to be divided into subgroups which show a consistent behavior for an entire set of points. Dividing the points into groups and generating a model for each group is called submodelling. This was the next approach used for a variety of groupings.

Table 2. Single Project and Product Model equations

COCOMO

Product: $Effort = 3.826 \times KWOM^{1.0}$

Project: $Effort = 4.088 \times KWOM^{1.0}$

COPMO

Product: $Effort = 47.953 + 0.279 \times KWOM + 2.017 \times \overline{P}^{1.675}$

Project: $Effort = 113.133 + 0.771 \times KWOM + 0.369 \times \overline{P}^{2.00}$

Putnam

Product: $Effort = 1.677 \times (KWOM^3 / Duration^4)$

Project: $Effort = 1.760 \times (KWOM^3 / Duration^4)$

Table 3. Results for Single Product and Single Project Models

		$\overline{Error}$	$\overline{RE}$	R^2	$\overline{MRE}$
COCOMO	Product:	196.5	-0.43	.66	0.85
	Project:	308.9	-0.56	.66	0.97
COPMO	Product:	116.0	-0.57	.93	0.86
	Project:	165.7	-0.93	.96	1.19
Putnam	Product:	295.8	0.91	.11	0.91
	Project:	477.4	0.90	.09	0.95

Language Submodels

Language is the first method used to divide the product points into groups. Products are coded in three languages (high order (HO), assembler (AS) and signal processing language (SL)) and thus were divided into three groups. The resulting division is twenty-six HO data points, sixteen SL data points, and thirteen AS data points. A best-fit correlation is performed to calculate the constants for the equations. The equations and results for the Language Submodels are shown in Table 4 and 5 respectively.

The results of the language submodels (Table 5) show slight improvement over the single product and project models. The improvement can be considered insignificant because the overall improvement was caused by each submodel having only a slight improvement for a segment of the data. However, the results for the HO data show a significantly lower $\overline{Error}$ even though the class consists of almost half of the data set. The SL data also shows definite improvement, even though the data set is small. The AS data does poorly by all measures. The poor performance of the AS models can be credited to the extreme range of the AS data. The AS data has ranges of KESLOC from 8 to 850, effort from 24 to 6201, and duration from 12 to 55. This variance is the greatest in the data. In an overall review, the language submodels seem to be capable of some accurate predictions, but are also prone to have large errors.

Table 4. Language Submodel Equations

HO

COCOMO: $Effort = 2.210 \times KESLOC^{1.0}$

COPMO: $Effort = 37.325 + 1.397 \times KESLOC + 10.208 \times \overline{P}^{1.0}$

Putnam: $Effort = 2.435 \times KESLOC^3 / Duration^4$

SL

COCOMO: $Effort = 3.180 \times KESLOC^{1.0}$

COPMO: $Effort = 45.262 + 1.992 \times KESLOC + 4.392 \times \overline{P}^{1.0}$

Putnam: $Effort = 3.731 \times KESLOC^3 / Duration^4$

AS

COCOMO: $Effort = 0.031 \times KESLOC^{1.81}$

COPMO: $Effort = 65.402 - 0.757 \times KESLOC + 4.264 \times \overline{P}^{1.56}$

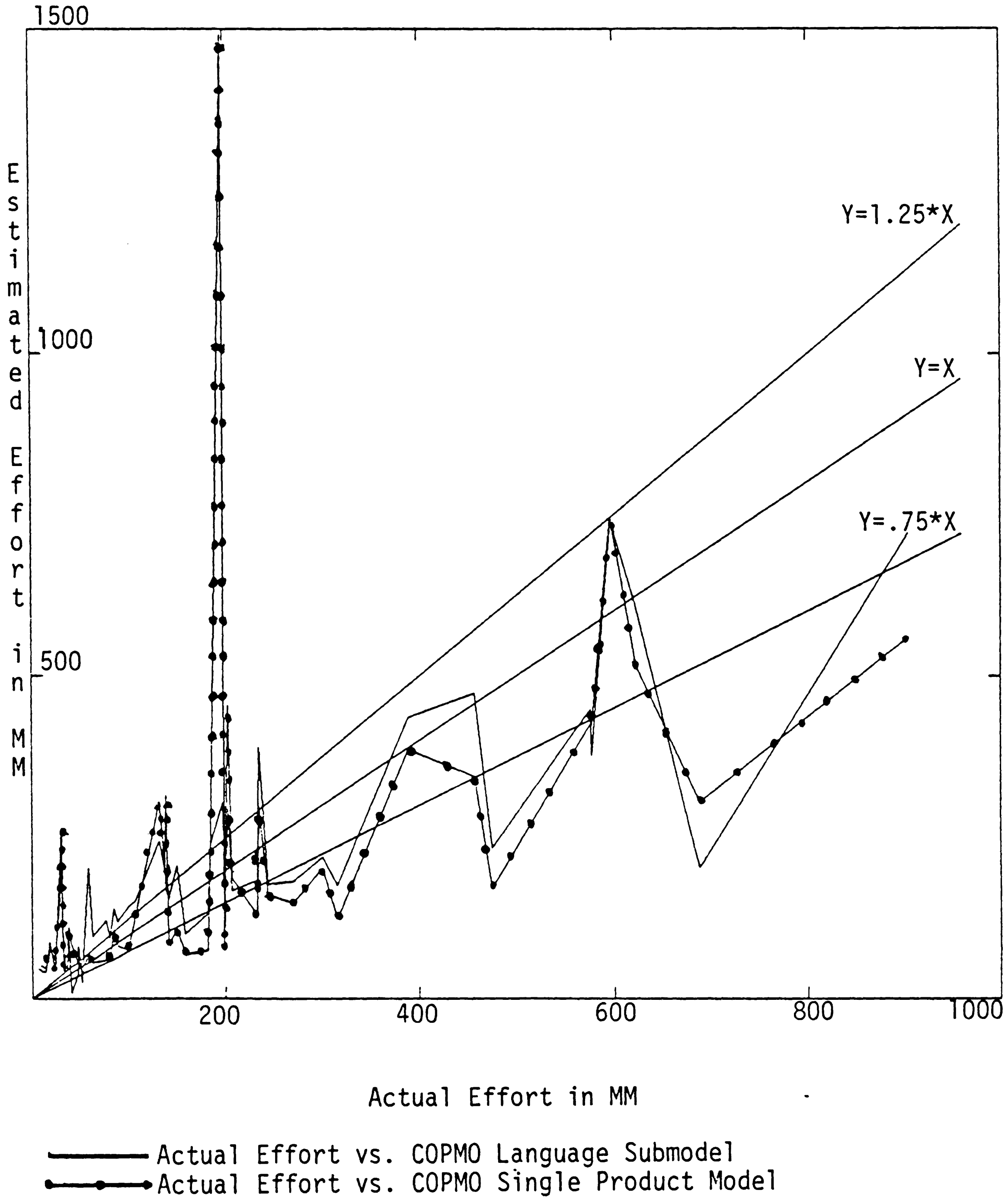
Putnam: $Effort = 42.821 \times KESLOC^3 / Duration^4$

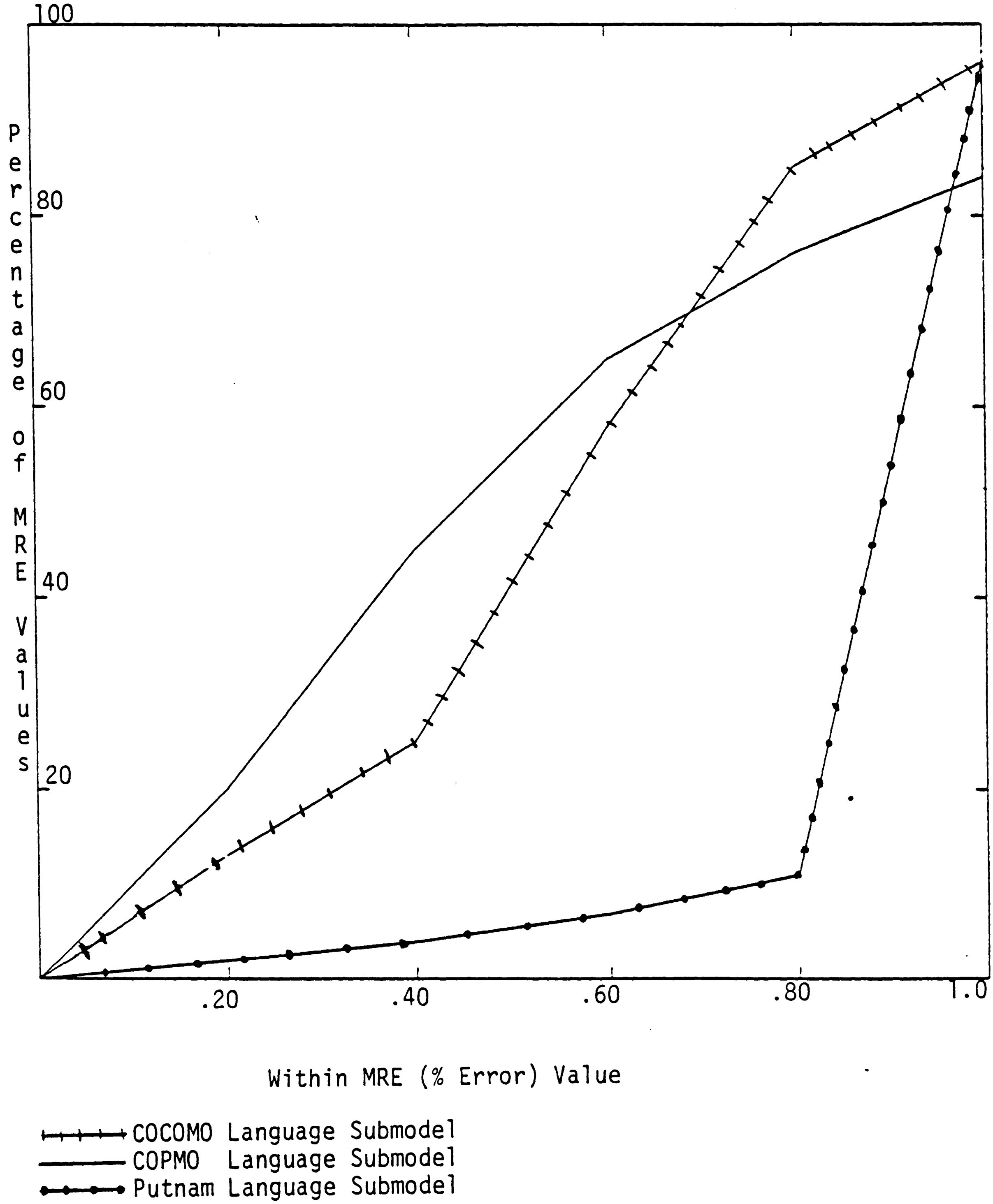
Table 5. Language Submodel results

		$\overline{Error}$	$\overline{RE}$	R^2	$\overline{MRE}$
HO	COCOMO:	91.5	.48	.90	0.51
	COPMO:	63.0	-.66	.94	0.85
	Putnam:	144.4	.95	.77	0.95
SL	COCOMO:	100.5	.06	.57	0.62
	COPMO:	91.2	-.67	.34	0.88
	Putnam:	161.0	.81	.08	0.92
AS	COCOMO:	215.3	-.02	.96	1.14
	COPMO:	70.8	-.01	.99	0.56
	Putnam:	676.8	.20	.49	1.21

The following graphs are specific representations of the results of the language submodels. The first graph is a plotting of the actual effort values against the predicted effort estimates generated by the COPMO model for the single product model and the language submodels. The graph also contains a Y=X line which represents estimates equal to actual values, a Y=1.25*X line that represents overestimates of 25%, and a Y=.75*X line that represents underestimates of 25%. The significance of Y=1.25*X line and the Y=.75*X line is that create a region of estimates within a 25% error. Estimates within a 25% error are considered acceptable by current modelling standards [Dem82]. Therefore, the ideal graph has all its estimates within the 25% region. Specifically, the graph shows the improvement in accuracy of the language submodel over the single product submodel. The improvement is shown by more points within the 25% error region. However, the majority points are still not in the region.

The second graph (called an MRE graph) plots the percent of language submodel MRE values within varied MRE levels. The information supplied by the MRE graph is the percentage of products within a 20%, 40%, 60%, 80% and 100% error. The information to be extracted from the MRE graph is that the COPMO model has the most products within a low error level. The COCOMO exhibits a moderate performance since most of its MRE values are between 20% and 40% error. The Putnam model might appear to do well, since almost 100% of its MRE values are within 100% error, however the model has only 10% of its MRE values within 80% error. The Putnam model generates such values by consistently underestimating.


1500
1000
500
Estimated Effort in MM
200
400
600
800
1000
Y=1.25*X
Y=X
Y=.75*X
Actual Effort in MM
Actual Effort vs. COPMO Language Submodel
Actual Effort vs. COPMO Single Product Model


100
80
60
40
20
Percentage of MRE Values
.20
.40
.60
.80
1.0
Within MRE (% Error) Value
COCOMO Language Submodel
COPMO Language Submodel
Putnam Language Submodel

Product SubModels

Product type is the next division used to generate submodels. The data is composed of five product types. The actual product type had been coded, so each group simply represented a unique set of points. The sizes of the groups were: 7, 19, 8, 9 and 12 respectively. All three models are calculated with a best-fit method for each of the the five groups. The equations and results are displayed in Tables 6 and 7 respectively.

Results from the product submodels show considerable improvement over language submodels and thus the single models. Another trend occurring in the product submodels, is the superiority of the predictions from the COPMO model. The COPMO model performed significantly better by all measures in all types other than type 4. The most significant result is the overall improvement in effort prediction. The use of product type as a submodel criteria has been successful elsewhere, as noted in many of the models reviewed in the first chapter. There are a variety of possible intuitive reasons for the success of product type submodelling. If a project is divided into commonly developed types of product, there is commonality between products of given type. The commonality of products causes more accurate size estimates as an estimator becomes more experienced. Another reason for the improved predictability is that generally the same personnel develop the same type of code for numerous projects. The same personnel will use similar techniques and styles, for multiple products with even slightly different requirements. Stratifying the development techniques and styles leads to a more predictable linear pattern of required effort.

Table 6. Product Submodel Equations

Type 1

COCOMO: $Effort = 2.628 \times KWOM^{1.07}$

COPMO: $Effort = 12.707 + 0.211 \times KWOM + 5.897 \times \overline{P}^{1.87}$

Putnam: $Effort = 82.791 \times KWOM^3 / Duration^4$

Type 2

COCOMO: $Effort = 5.677 \times KWOM^{1.2}$

COPMO: $Effort = -63.772 + 5.252 \times KWOM + 6.586 \times \overline{P}^{1.0}$

Putnam: $Effort = 13.016 \times KWOM^3 / Duration^4$

Type 3

COCOMO: $Effort = 0.029 \times KWOM^{1.82}$

COPMO: $Effort = 120.419 - 1.206 \times KWOM + 4.467 \times \overline{P}^{1.56}$

Putnam: $Effort = 4.541 \times KWOM^3 / Duration^4$

Type 4

COCOMO: $Effort = 2.445 \times KWOM^{1.02}$

COPMO: $Effort = 24.509 + 2.473 \times KWOM - 0.134 \times \overline{P}^{1.0}$

Putnam: $Effort = 3.148 \times KWOM^3 / Duration^4$

Type 5

COCOMO: $Effort = 1.184 \times KWOM^{1.22}$

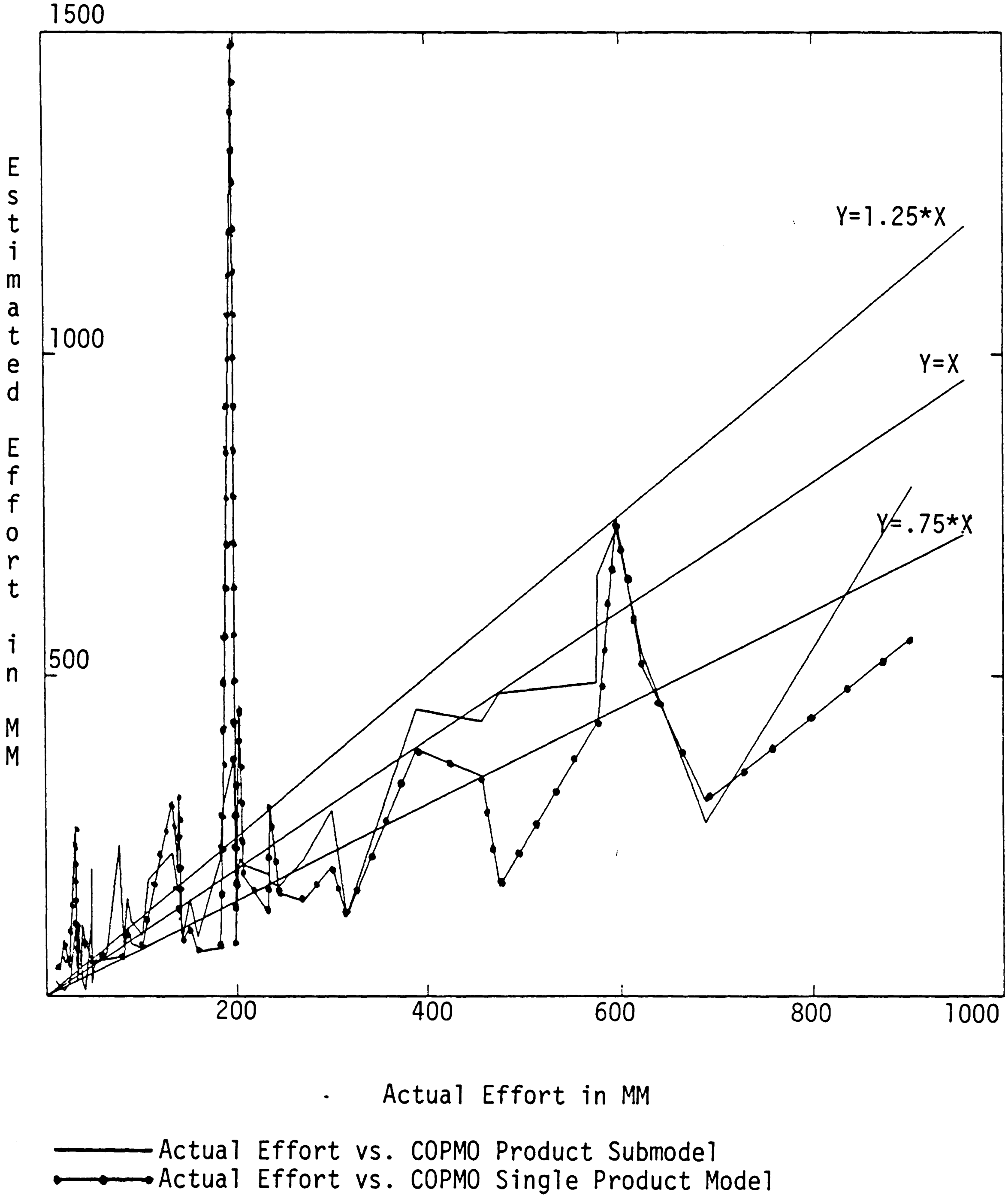
COPMO: $Effort = -0.933 + 0.602 \times KWOM + 20.353 \times \overline{P}^{1.0}$

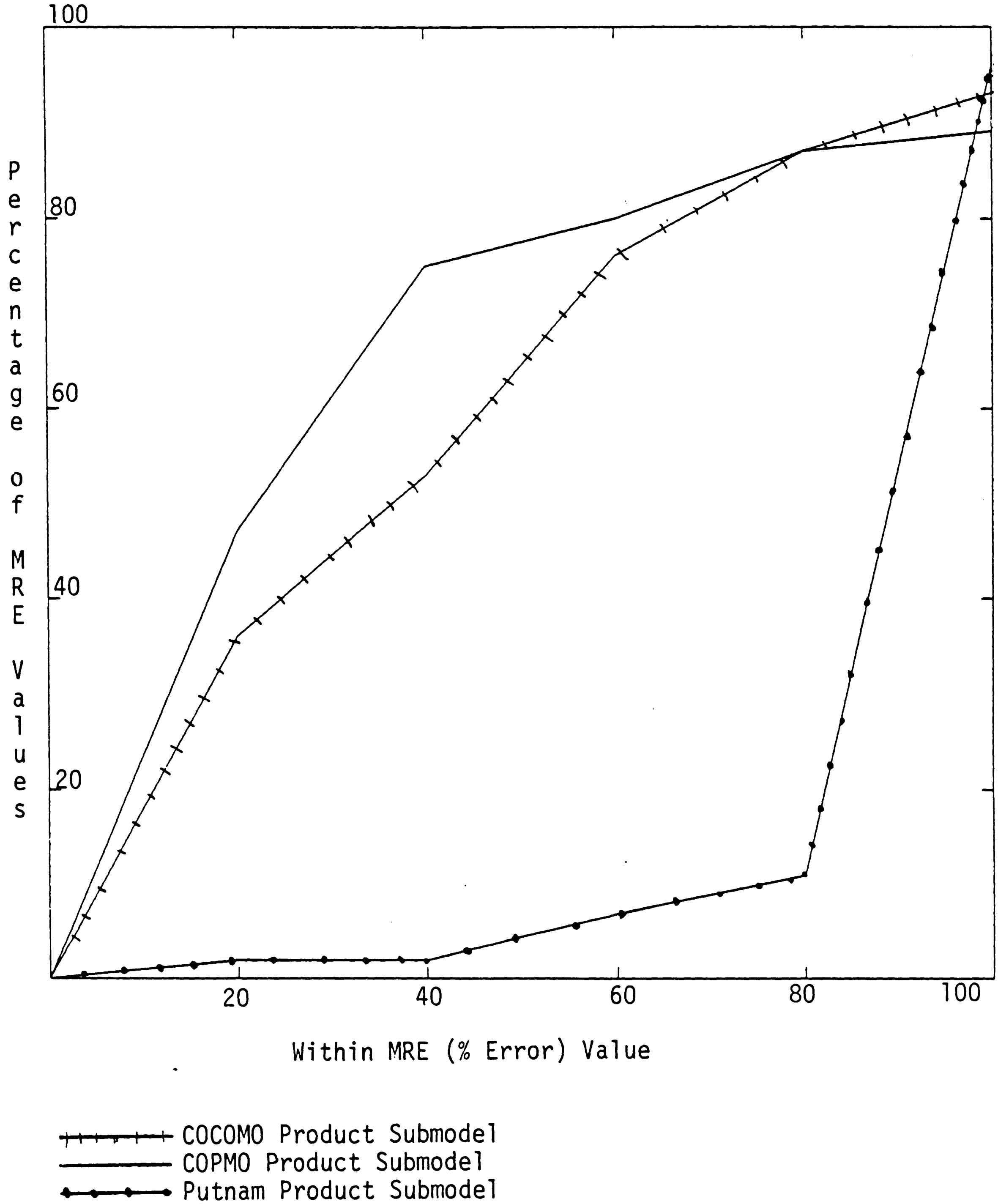
Putnam: $Effort = 1.410 \times KWOM^3 / Duration^4$

Table 7. Product Submodel results

Type	Model	$\overline{Error}$	$\overline{RE}$	R^2	$\overline{MRE}$
1	COCOMO	61.3	-.28	.67	0.74
	COPMO	5.1	.19	.99	0.35
	Putnam	90.6	.74	.04	0.94
2	COCOMO	87.0	-.36	.79	0.63
	COPMO	78.4	-.29	.57	0.53
	Putnam	216.4	.87	.06	0.93
3	COCOMO	171.0	-.48	.98	1.35
	COPMO	72.8	-.09	.99	0.97
	Putnam	923.8	-.15	.50	1.49
4	COCOMO	37.0	.13	.89	0.29
	COPMO	38.1	-.21	.80	0.33
	Putnam	131.6	.79	.12	0.91
5	COCOMO	110.3	.49	.94	0.50
	COPMO	40.8	-.09	.99	0.25
	Putnam	174.3	.91	.84	0.91

The following effort graph exhibits the results of Table 7. The variance of product submodel estimates is far less than the variance of the single product model. The MRE graph shows improved performance by both the COPMO and the COCOMO submodels. The performance of the COPMO model is most significant since it has almost 80% of the products within a 40% error.


1500
1000
500
200
400
600
800
1000
Y=1.25*X
Y=X
Y=.75*X
Estimated Effort in MM
Actual Effort in MM
Actual Effort vs. COPMO Product Submodel
Actual Effort vs. COPMO Single Product Model



100
80
60
40
20
Percentage of MRE Values
20
40
60
80
100
Within MRE (% Error) Value
COCOMO Product Submodel
COPMO Product Submodel
Putnam Product Submodel

Size SubModel

The third submodelling approach is the use of size. The sizes of products are divided into classes by creating logical ranges of size in the hope that the classes might capture types of behavior. The ranges that will be used are: $KWOM < 10$, $10 \leq KWOM < 30$, $30 \leq KWOM < 60$, $60 \leq KWOM < 100$, and $KWOM \geq 100$. The number of data points in each was 10, 16, 14, 8, and 7 respectively. All three models were generated for each class using a best-fit method to generate the constants. The equations are in Table 8 and the results are in Table 9.

Size submodelling produces results better than expected. The success of the product submodels can be credited to the fact that five model equations can be more accurate than three or one. However, the size submodels significantly reduce error even further than the product submodels while using the same number of model equations. The COPMO model is again the overall best predictor. The COCOMO model also performs well in all classes, other than class 4. The justification for the success of this submodelling is not obvious. A possible explanation of this success is that projects of a given size class may be developed in consistent manners. Examples of this could be that the small projects (class 0) are developed by the same individuals, or individual. Another theory is that totally different design and development techniques are used on the largest products (class 4) in comparison to medium sized products (class 2).

The size submodel produces an effort graph which has the majority of its values within 25% of the actual value. The MRE graph confirms the accuracy of the COPMO size submodels, as shown by 80% of the products within 40% error, and 90% of the products within 60% error. The COCOMO size submodel performs fairly well, and the Putnam size submodel exhibits its typical poor performance.

Table 8. Size Submodel Equations

Class 0 (KWOM < 10)

COCOMO: $Effort = 4.842 \times KWOM^{1.0}$

COPMO: $Effort = -3.891 + 4.795 \times KWOM + 1.280 \times \bar{P}^{1.0}$

Putnam: $Effort = 13.491 \times KWOM^{3} / Duration^{4}$

Class 1 (10 <= KWOM < 30)

COCOMO: $Effort = 2.711 \times KWOM^{1.2}$

COPMO: $Effort = -3.722 + 3.710 \times KWOM + 3.279 \times \bar{P}^{1.0}$

Putnam: $Effort = 5.811 \times KWOM^{3} / Duration^{4}$

Class 2 (30 <= KWOM < 60)

COCOMO: $Effort = 5.811 \times KWOM^{1.0}$

COPMO: $Effort = 115.440 - 0.759 \times KWOM + 5.094 \times \bar{P}^{1.56}$

Putnam: $Effort = 60.629 \times KWOM^{3} / Duration^{4}$

Class 3 (60 <= KWOM < 100)

COCOMO: $Effort = 0.051 \times KWOM^{2.0}$

COPMO: $Effort = -640.502 + 11.229 \times KWOM + 0.420 \times \bar{P}^{2.0}$

Putnam: $Effort = 2.676 \times KWOM^{3} / Duration^{4}$

Class 4 (KWOM >= 100)

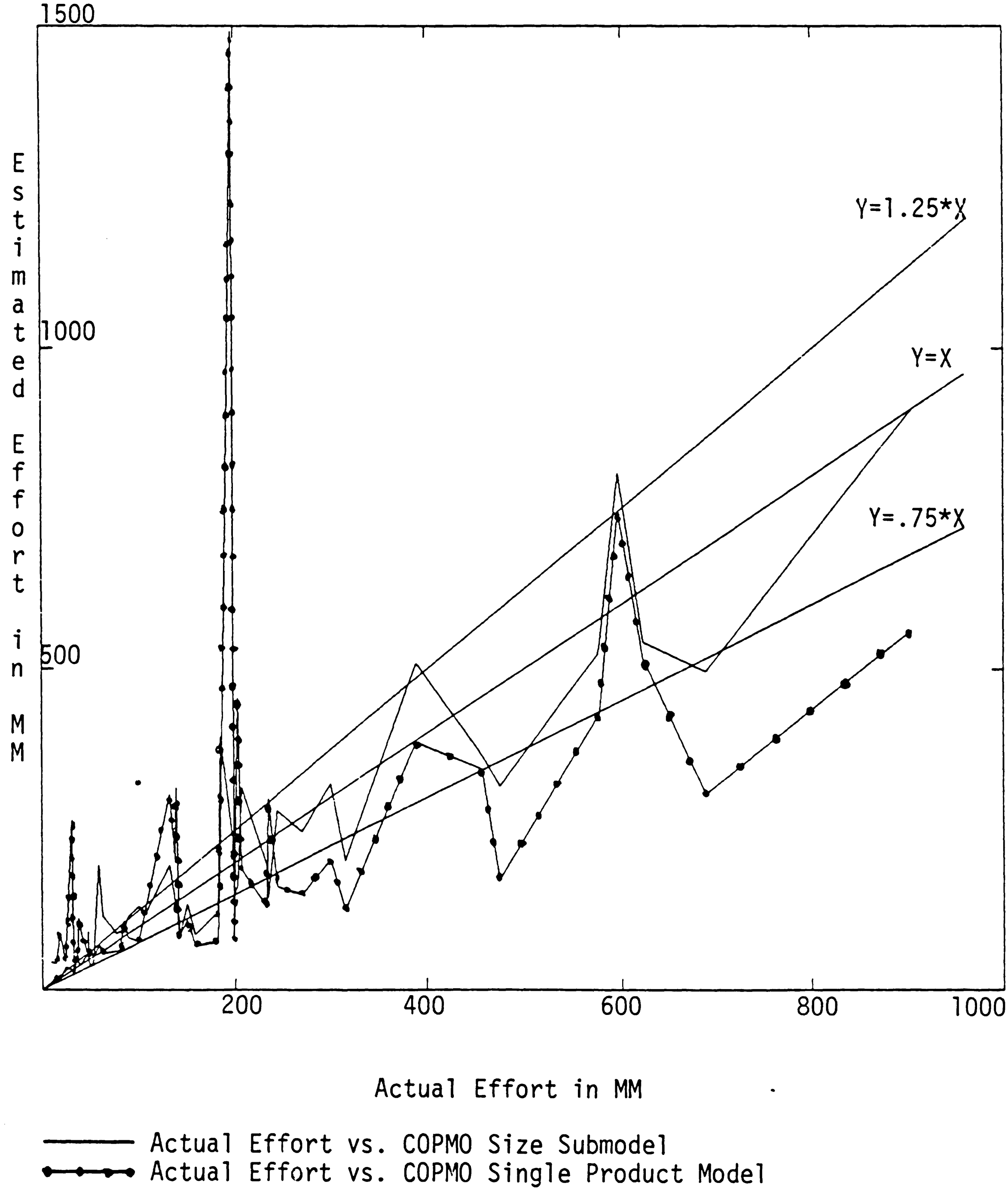
COCOMO: $Effort = 0.839 \times KWOM^{1.22}$

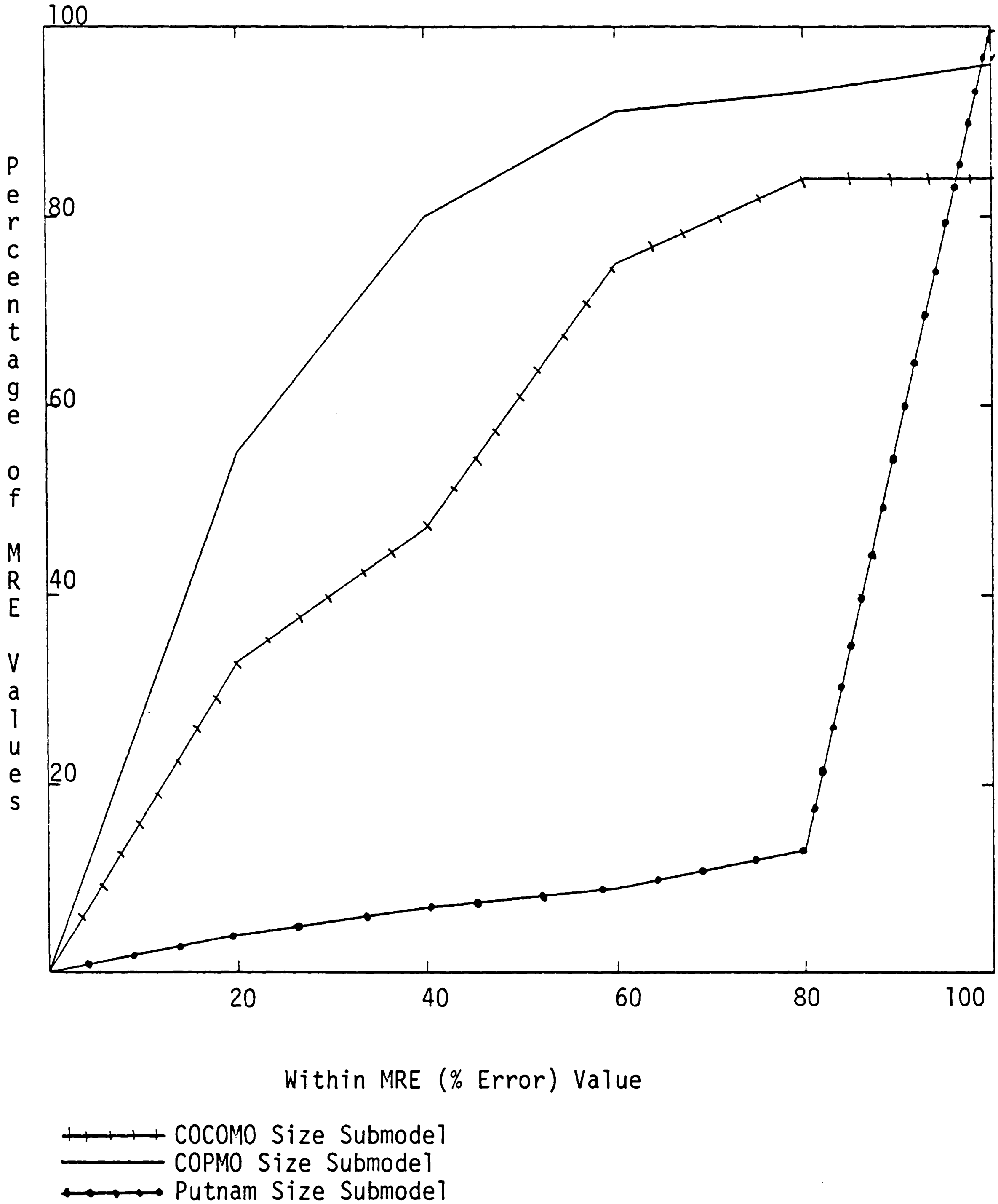
COPMO: $Effort = 158.640 - 0.699 \times KWOM + 3.793 \times \bar{P}^{1.58}$

Putnam: $Effort = 1.670 \times KWOM^{3} / Duration^{4}$

Table 9. Size Submodel results

Class	Model	$\overline{Error}$	$\overline{RE}$	R^2	$\overline{MRE}$
0	COCOMO	7.2	-.05	.92	0.30
	COPMO	4.7	-.05	.70	0.19
	Putnam	25.0	.84	.16	0.89
1	COCOMO	40.6	-.21	.82	0.42
	COPMO	25.1	-.11	.67	0.24
	Putnam	93.3	.90	.22	0.92
2	COCOMO	158.4	-.65	.53	0.93
	COPMO	67.2	-.14	.84	0.34
	Putnam	266.4	.83	.04	0.88
3	COCOMO	123.9	-.92	.79	1.10
	COPMO	80.1	-.24	.78	0.54
	Putnam	238.4	.82	.04	0.90
4	COCOMO	917.3	-1.40	.65	1.84
	COPMO	60.0	-.01	.99	0.11
	Putnam	1216.4	.77	.11	0.83


1500
1000
500
Estimated Effort in MM
200
400
600
800
1000
Y=1.25*X
Y=X
Y=.75*X
Actual Effort in MM
Actual Effort vs. COPMO Size Submodel
Actual Effort vs. COPMO Single Product Model


100
80
60
40
20
Percentage of MRE Values
20
40
60
80
100
Within MRE (% Error) Value
COCOMO Size Submodel
COPMO Size Submodel
Putnam Size Submodel

Duration SubModels

The final approach to submodelling is to divide up the products by the expected duration of the product development. These classes attempt to capture logical lengths of development similar to the classes of the size submodels. The classes are as follows: $Elapse < 10$ (*months*), $10 \le Elapse < 20$, $20 \le Elapse < 30$, $30 \le Elapse < 40$, and $Elapse \ge 40$. The number of datapoints for each class was: 8, 10, 19, 11, and 7 respectively. Once again the constants were generated with a best-fit method for each of the three models for all five classes. The equations and results are shown in Tables 10 and 11 respectively.

The results in Table 11 are once again an improvement on the previous method. Duration Submodelling shows consistently good prediction for both the COCOMO and COPMO models. The Putnam model performs far better than normal. The COPMO model is unquestionably the best predictor. Overall, the COPMO model predicted with an error of just over 20 MM or an error of about 18%. This performance is consistent with any of the best performance claimed in literature. The only drawback of this model is the need for fairly accurate prediction of not only size but also duration. However, given the accuracy of the model and the fairly broad sizes of the classes it is a model worth further investigation. The apparent reason for its success could be simply that a given amount of code can be done over a period of time regardless of language or type.

Table 10. Duration Submodel Equations

Class 0 (Elapse < 10)

COCOMO: $Effort = 3.670 \times KWOM^{1.0}$

COPMO: $Effort = 3.783 + 1.537 \times KWOM + 3.710 \times \overline{P}^{1.0}$

Putnam: $Effort = 2.999 \times KWOM^{3} / Duration^{4}$

Class 1 (10 < = Elapse < 20)

COCOMO: $Effort = 4.624 \times KWOM^{1.0}$

COPMO: $Effort = 25.130 + 1.346 \times KWOM + 1.028 \times \overline{P}^{1.8}$

Putnam: $Effort = 58.108 \times KWOM^{3} / Duration^{4}$

Class 2 (20 < = Elapse < 30)

COCOMO: $Effort = 3.356 \times KWOM^{1.0}$

COPMO: $Effort = 12.076 - 0.301 \times KWOM + 25.441 \times \overline{P}^{1.0}$

Putnam: $Effort = 8.926 \times KWOM^{3} / Duration^{4}$

Class 3 (30 < = Elapse < 40)

COCOMO: $Effort = 1.729 \times KWOM^{1.0}$

COPMO: $Effort = 7.469 - 0.193 \times KWOM + 34.386 \times \overline{P}^{1.0}$

Putnam: $Effort = 1.408 \times KWOM^{3} / Duration^{4}$

Class 4 (Elapse > = 40)

COCOMO: $Effort = 3.716 \times KWOM^{1.1}$

COPMO: $Effort = -56.491 + 1.441 \times KWOM + 36.945 \times \overline{P}^{1.04}$

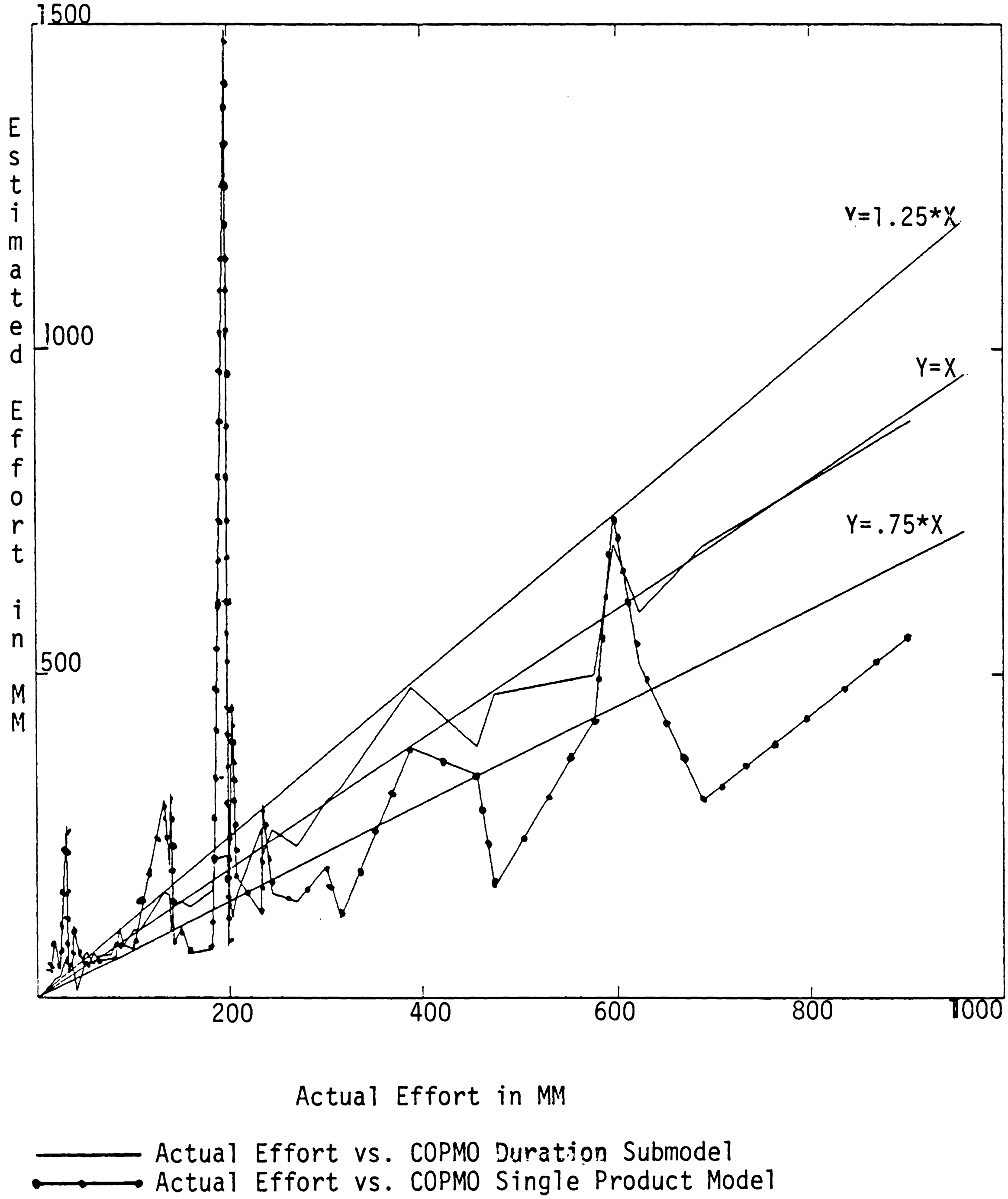
Putnam: $Effort = 92.381 \times KWOM^{3} / Duration^{4}$

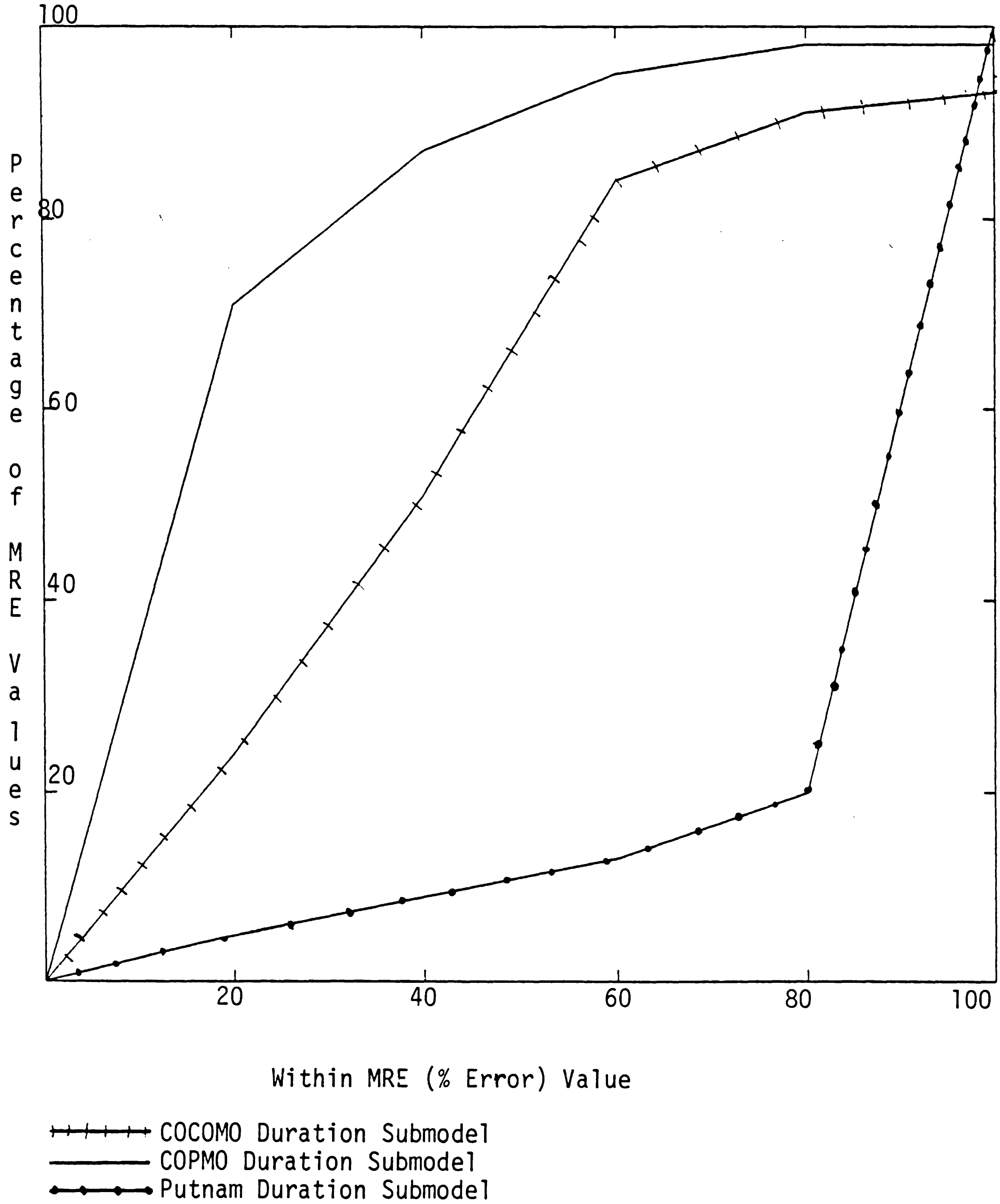
Table 11. Duration Submodel results

Class	Model	$\overline{Error}$	$\overline{RE}$	R^2	$\overline{MRE}$
0	COCOMO	52.9	.27	.66	0.47
	COPMO	27.6	-.26	.78	0.42
	Putnam	68.9	.68	.44	0.80
1	COCOMO	29.6	-.07	.89	0.22
	COPMO	13.6	-.03	.96	0.11
	Putnam	84.7	.68	.43	0.77
2	COCOMO	106.3	-.20	.83	0.62
	COPMO	30.3	-.08	.96	0.15
	Putnam	210.1	.90	.25	0.92
3	COCOMO	132.8	-.21	.81	0.98
	COPMO	11.9	-.02	.99	0.19
	Putnam	151.6	.90	.79	0.90
4	COCOMO	130.6	-.72	.99	1.01
	COPMO	11.7	-.04	.99	0.10
	Putnam	269.1	.83	.97	0.83

The accuracy of the COPMO duration submodel is confirmed by the Effort graph and the MRE graph. The values in the Effort graph hover close to the X = Y line and have only one value outside the 25% area. The MRE graph displays almost 90% of the MRE values within 40% error and 95% of the MRE values within 60% error for the COPMO duration submodel.

An overall review of the preliminary results show a great range of predicting accuracy. It is apparent that the submodelling approach is more successful than the single model approach. At present, the best results are exhibited by the duration submodel, however looking at $\overline{Error}$ for the COPMO submodels in table 11, it is obvious that error is as large as fifty for some products. So even though the duration submodel performs well there, is room for improvement.


1500
1000
500
Estimated Effort in MM
200
400
600
800
1000
Y=1.25*X
Y=X
Y=.75*X
Actual Effort in MM
Actual Effort vs. COPMO Duration Submodel
Actual Effort vs. COPMO Single Product Model


Percentage of MRE Values
100
80
60
40
20
20
40
60
80
100
Within MRE (% Error) Value
COCOMO Duration Submodel
COPMO Duration Submodel
Putnam Duration Submodel

Chapter 4

Adjustment Multipliers

The models presented in the previous chapter demonstrated varying levels of accuracy, however all models have some data points in the set where the model generates a poor estimate. Adjustment multipliers are a method which attempts to correct the poor predictions, yet not affect the good predictions.

To summarize adjustment multipliers, the method is a refinement of an effort estimate based on a product characteristic. As a general example, for a product requiring an effort estimate, the size and average personnel are used in a COPMO equation (selected using a product characteristic), to generate an effort estimate. A set of adjustment multipliers is selected (based on another product characteristic, product type, language, size or duration), the effort estimate (from the submodel), is multiplied by a multplier from the set (based on the adjustment multiplier characteristic) to generate the final estimate. The adjustment multipliers are to be used in combination with the effort estimates generated by only the COPMO equations due to their superiority.

A detailed example will now be provided to precisely describe the adjustment multiplier method. The example product has the following characteristics: expected KWOM = 28, expected $\overline{P} = 14$, expected duration is between 10 and 20 months, language will be HO, and product type is 3. The duration submodels and the size adjustment multipliers will be used. Since the expected duration is between 10 and 20 months the class 1 COPMO submodel is the equation used to generate the initial estimate.

$$COPMO\text{:} \quad Effort = 25.130 + 1.346 \times KWOM + 1.028 \times \overline{P}^{1.8} \quad \text{(from Table 8)}$$

Using KWOM = 28 and $\overline{P} = 14$, the equation produces an initial effort estimate of 182 MM. Next, the size adjustment multiplier is selected (from Table 18), the adjustment multiplier class is 1 (KWOM is between 10 KWOW and 30 KWOM), so the appropriate multiplier is 1.032. Therefore the final effort estimate is 182 × 1.032 which is equal to 188 MM.

The adjustment multipliers are generated by doing a best-fit correlation between the estimated values of the submodels and the actual effort values. The following sets of multipliers are those which would be used on an estimate generated from the language submodels.

Table 12. Product Adjustment Multipliers for Language Submodel Est.

Type 1: Effort = 1.140 * Lang. Effort Est.

Type 2: Effort = 1.054 * Lang. Effort Est.

Type 3: Effort = 1.000 * Lang. Effort Est.

Type 4: Effort = 0.839 * Lang. Effort Est.

Type 5: Effort = 0.990 * Lang. Effort Est.

Table 13. Results of Product Adjustment Multipliers

	$\overline{Error}$	$\overline{RE}$	R^2	$\overline{MRE}$
Language Submodel	71.8	-.56	.98	0.77
Product Adjustment Multipliers	71.2	-.56	.99	0.77

Table 14. Duration Adjustment Multipliers for Language Submodel Est.

Class 0: (Elapse < 10)
Effort = 0.573 * Language Effort Estimate

Class 1: (10 < = Elapse < 20)
Effort = 0.697 * Language Effort Estimate

Class 2: (20 < = Elapse < 30)
Effort = 0.997 * Language Effort Estimate

Class 3: (30 < = Elapse < 40)
Effort = 1.014 * Language Effort Estimate

Class 4: (Elapse > = 40)
Effort = 1.005 * Language Effort Estimate

Table 15. Results of Duration Adjustment Multipliers

	$\overline{Error}$	$\overline{RE}$	R^2	$\overline{MRE}$
Language Submodel	71.8	-.56	.98	0.77
Duration Adjustment Multipliers	58.5	-.26	.99	0.52

Table 16. Size Adjustment Multipliers for Language Submodel Est.

Class 0: (KWOM < 10)
Effort = 0.292 * Language Effort Estimate

Class 1: (10 <= KWOM < 30)
Effort = 0.675 * Language Effort Estimate

Class 2: (30 <= KWOM < 60)
Effort = 1.339 * Language Effort Estimate

Class 3: (60 <= KWOM < 100)
Effort = 0.967 * Language Effort Estimate

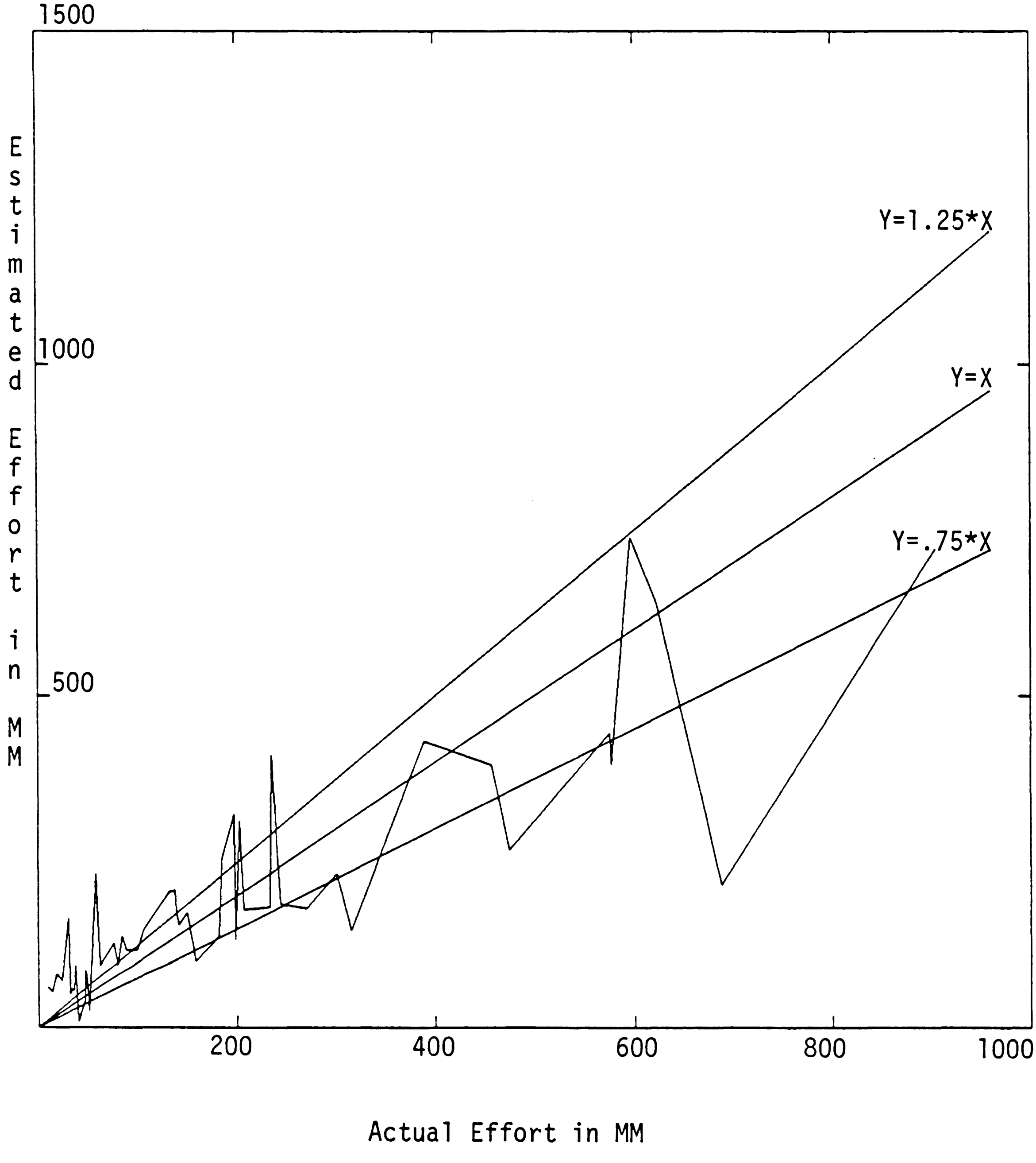
Class 4: (KWOM >= 100)
Effort = 0.999 * Language Effort Estimate

Table 17. Results of Size Adjustment Multipliers

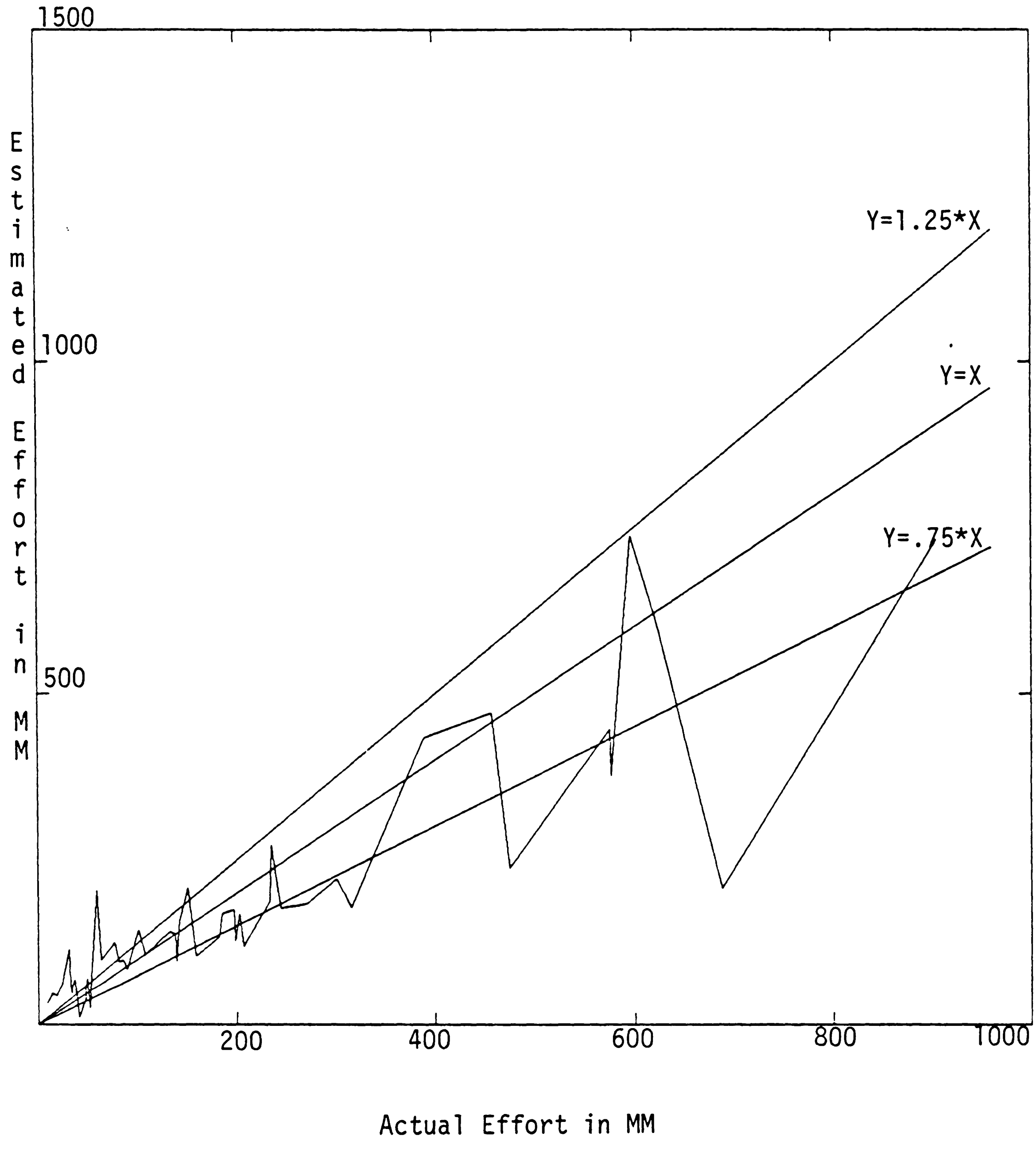
	$\overline{Error}$	$\overline{RE}$	R^2	$\overline{MRE}$
Language Submodel	71.8	-.56	.98	0.77
Duration Adjustment Multipliers	46.5	-.05	.99	0.32

The overall set of adjustment multipliers for the language estimates are a good example of the effects of adjustment multipliers in general. The results obtained using product adjustment multipliers (Table 13) are only a slight improvement over the language submodel results. The minimal improvement infers that product type is really not a worthwhile second criteria for language submodel estimates. The duration adjustment multipliers showed a signifcant improvement the language submodel results in all measurements (Table 15). The equations for class 0 and class 1 (Table 14) caused the reduction in error. The other equations of the duration adjustment multipliers have little effect as shown by closeness of their constants to one. The size adjustment multipliers produce an even more significant set of results (Table 17). For this set of the multipliers (Table 16) class 0, 1 and 2 produces the reduction in effort. The trend of lower (size or duration) class equations performing the reductions in error is a common one. The implications of the trend are that products of a small duration or size tend to be overestimated. The adjustment multiplier is a method which can compensate for that model weakness.

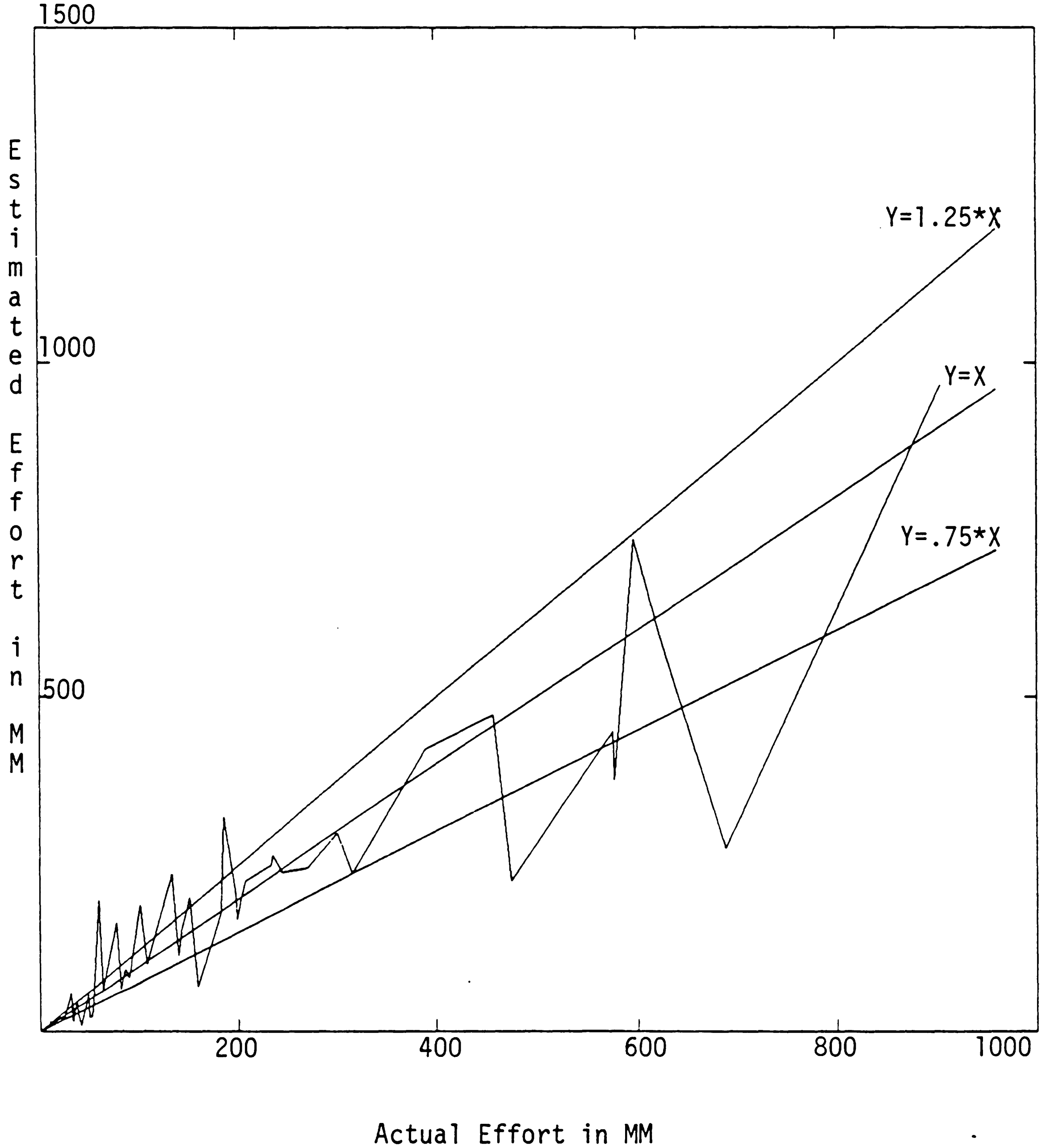
The three following effort graphs show the effects of the adjustment models. To observe the improvement in estimation the graphs should be compared to their submodel counterpart. As expected the product adjustment multipliers have little effect. The duration adjustment multipliers move many of the low effort values closer to the X = Y line. The size adjustment multipliers appear to have little effect, but many of the larger effort values are closer to the X = Y line.



Product Adjustment Multipliers on Language Submodel Estimates



Duration Adjustment Multipliers on Language Submodel Estimates



Size Adjustment Multipliers on Language Submodel Estimates

One other set of adjustment multipliers worth noting, is the size adjustment multipliers on the duration submodel estimates. The equations and results for this set of size adjustment multipliers are supplied in Tables 18 and 19. The significance of the size adjustment multipliers used on the duration submodel estimates are that they represent the highest accuracy the study achieved. $\overline{MRE}$ represents the percent error, the $\overline{MRE}$ value of 13% is a highly respectable value since traditionally classic models have a performance of 15% - 20% error [Dem82]. Although the other measurements in Table 19 are not significantly different other than $\overline{MRE}$, the improvement is worth noting since the improvement was done on a highly accurate submodel.

The supplied effort graph shows the performance of the size adjustment multipliers on the duration submodel estimates. The graph displays the line which has the least variance from the X = Y line.

The adjustment multipliers for estimates generated by the product submodels, the size submodels and the duration submodels can be found in Appendix A.

Table 18. Size Adjustment Multipliers for Duration Submodel Est.

Class 0: (KWOM < 10)
Effort = 0.689 * Duration Effort Estimate

Class 1: (10 < = KWOM < 30)
Effort = 1.032 * Duration Effort Estimate

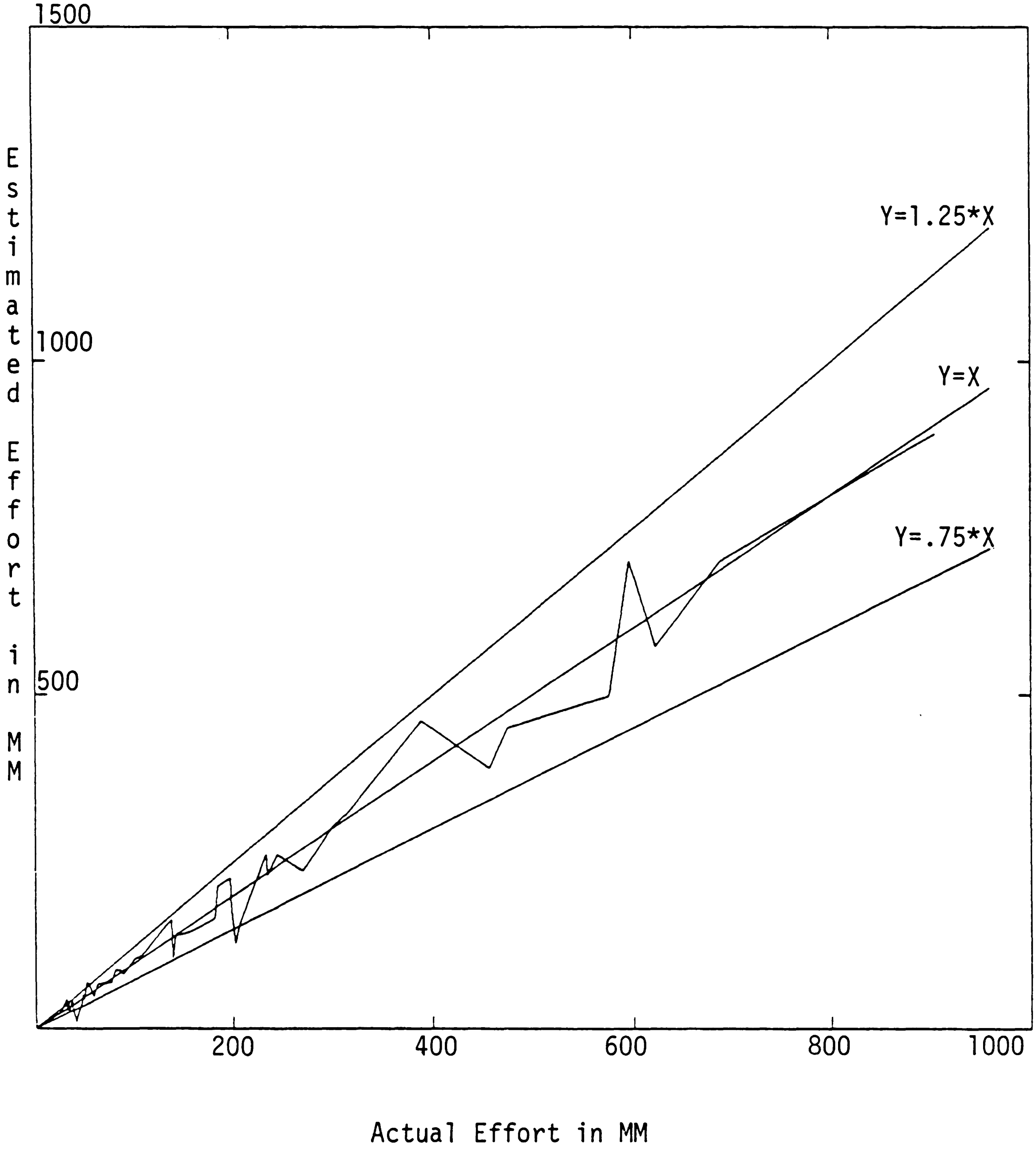
Class 2: (30 < = KWOM < 60)
Effort = 1.003 * Duration Effort Estimate

Class 3: (60 < = KWOM < 100)
Effort = 0.962 * Duration Effort Estimate

Class 4: (KWOM > = 100)
Effort = 1.001 * Duration Effort Estimate

Table 19. Results of Size Adjustment Multipliers on the Duration Submodel

	$\overline{Error}$	$\overline{RE}$	R^2	$\overline{MRE}$
Duration Submodel	20.9	-.08	.99	0.18
Size Adjustment Multipliers	20.0	-.003	.99	0.13



Size Adjustment Multipliers on Duration Submodel Estimates

Chapter 5

Constructive Methods

A comparison of the results of the single product model versus the single project model (Table 3) reveals that estimation done at product level is more accurate than at the project level. However, most contracted software is generally of a size requiring subdivision of the project into multiple products. Product estimating models have limited accuracy, therefore effort estimates can be underestimates as easily as overestimates. If product effort estimates were added up to form a project estimate, there can be a cancelling effect to reduce the error in the original estimate. Summing up product estimates to form a project estimate is called the Constructive Modelling Technique (CMT).

An example of applying the constructive modelling technique follows. Given a project Z which is composed of two products X and Y, estimates are generated for for both products. The effort estimate for product X is 10 MM low, and the effort estimate for product Y is 10 MM high. If the products are developed in two seperate environments (i.e. subcontractors) the overall error is 20 MM. However, if both products are developed in a cooperative environment, and a constructive effort estimate is generated, the error would be zero. Although this is a contrived example it shows the potential benefit.

To test the CMT, the data should include a number of projects which contain a varying number of products. The IBM FSD dataset had a few multi-product projects, with varying numbers of products. The number of multi-product projects, although small, seemed sufficient to attempt the technique. The results are displayed in Table 20 which uses summed absolute error (SAE) which is defined as:

$$SAE = \sum_{i=1}^{n} |Actual_i - Estimate_i|$$

Table 20. Results of the Constructive Modelling Technique

Submodel	SAE	$SAEw/CMT$	$\overline{RE}$	$\overline{RE}w/CMT$	$\overline{MRE}$	$\overline{MRE}w/CMT$
Language	3950	3441	-.54	.40	0.76	0.64
Product Type	2977	2625	-.23	.20	0.47	0.45
Size	2462	2013	-.11	.04	0.28	0.21
Duration	1145	980	-.08	.06	0.19	0.19

Results specific to only the projects which contain multiple products are shown in Table 21. The results of Table 21 represent six projects consisting of a total twenty-seven products. In each case of the submodel estimates being added together to form a set of project estimates, the SAE was noticeably decreased. Although there is only a small number of projects to test the method, the consistent improvement by the CMT shows its potential value. The results also show that the poorer the original estimator is, the more effective the CMT is. The probable reason for this relation is greater room for cancelling of estimate error in the less accurate models. The only drawback of the results is the small number of multi-product projects.

Table 21. Results of CMT on Multi-product Projects

Submodel	SAE	$\overline{SAE}$ w/CMT
Language	1643	1134
Product Type	1088	736
Size	972	533
Duration	327	162

Chapter 6

Duration

The primary area of work in this study is the prediction of effort. The developed models are considered accurate, but still can generate predictions with considerable errors. Work in the areas of duration and risk is an attempt to create more information for the predictor, and to give the predictor information that can be used to verify effort predictions.

Given that a COPMO effort prediction is based on size and $\overline{P}$, predicted duration can be used to verify the calculated effort. Assuming the estimated duration is reasonably accurate, effort can be divided by duration, yielding a predicted $\overline{P}$. The predicted $\overline{P}$ value can then be compared with the $\overline{P}$ value used in the effort equation. If there is a significant difference between the two $\overline{P}$ values, the predicted $\overline{P}$ can be used in the effort equation in an iterative fashion to generate a new effort estimate.

The ultimate goal is to accurately predict duration. Predicting duration is theoretically no different than predicting effort. This model can use any available product characteristics to generate the estimate. The approach being taken by this study is the use of model formats based on size, $\overline{P}$, or effort in the single model format and in the multiple (submodelling technique) model format. The first format is the duration model developed by Putnam [Put77]. The model equation is as follows where a is calculated with a best-fit method.

$$Duration = a \times (\frac{Size^{.6}}{P^{.2}})$$

The second type of duration predicting equation is the size-based equation. This type uses size as the independent variable and differs in whether or not the equation contains an intercept point. The format of the two size-based equations are as follows, where a, b, and c are calculated with a best-fit method.

$$Duration = a \times Size^{b}$$

$$Duration = a + b \times Size^{c}$$

Effort-based equations are the third type of duration predicting equations. These equations use effort (MM) as the independent variable and also differ in whether or not the equation contains an intercept point. The constants of equations a, b, and c are calculated with a best-fit method. The format of equations is as follows.

$$Duration = a \times MM^{b}$$

$$Duration = a + b \times MM^{c}$$

The single model format is the first approach at modelling duration. Results from the single models are shown in Table 22. The results might appear to be favorable due to the low SAE values. However, SAE values above 500, and $\overline{Error}$ values greater than nine months are not close to the required accuracy. The inability to predict duration with single models is also reflected with poor R^2, $\overline{RE}$, and $\overline{MRE}$ values.

Table 22. **Single Equation Duration Models and Results**

1) Putnam's Duration Model

$$Duration = 2.719 \times \left(\frac{KWOM^{.6}}{\overline{P}^{.2}}\right)$$

$\underline{SAE} = 589$ $\quad$ $\underline{R^2} = .72$

$\overline{RE} = .13$ $\quad$ $\overline{MRE} = .42$

2) Size-based Duration Model without Intercept

$$Duration = 17.402 \times KWOM^{.1}$$

$\underline{SAE} = 499$ $\quad$ $\underline{R^2} = .83$

$\overline{RE} = -.59$ $\quad$ $\overline{MRE} = .82$

3) Size-based Duration Model with Intercept

$$Duration = 11.469 + 4.469 \times KWOM^{.3}$$

$\underline{SAE} = 504$ $\quad$ $\underline{R^2} = .19$

$\overline{RE} = -.53$ $\quad$ $\overline{MRE} = .77$

4) Effort-based Duration Model without Intercept

$$Duration = 15.876 \times MM^{.1}$$

$\underline{SAE} = 507$ $\quad$ $\underline{R^2} = .82$

$\overline{RE} = -.63$ $\quad$ $\overline{MRE} = .86$

5) Effort-based Duration Model with Intercept

$$Duration = 22.556 + 0.006 \times MM^{1.0}$$

$\underline{SAE} = 493$ $\quad$ $\underline{R^2} = .16$

$\overline{RE} = -.62$ $\quad$ $\overline{MRE} = .87$

The single model's inability to predict duration encourages the use of the submodelling technique. Again the data most likely represents multiple patterns of behavior rather than a single pattern. The use of the submodelling technique on the duration models is equivalent to the use with the effort models. In review, the data points are divided into groups based on a product characteristic (size, language or product type). An equation is then generated for each group using a best-fit method.

Applying each submodelling criteria, language, product type, and size to the five formats of duration model demonstrates that size is the only significant submodelling criteria. The equations and results are shown in Tables 23, 24 and 25 respectively. Although size is the best submodelling criteria, the improvement in the results over the single models is not very large. The SAE values have dropped well into the 400 range, but this is still far from the accuracy desired. The inaccuracy of the submodels is shown by the high $\overline{MRE}$ values.

Table 23. Size Submodel with Putnam's Duration Model

Overall Results

SAE = 443 $\overline{MRE} = .52$

Submodel Equations

$$\text{Class 0: } Duration = 7.590 \times \left(\frac{KWOM^{.6}}{\overline{P}^{.2}}\right)$$

$$\text{Class 1: } Duration = 4.328 \times \left(\frac{KWOM^{.6}}{\overline{P}^{.2}}\right)$$

$$\text{Class 2: } Duration = 4.462 \times \left(\frac{KWOM^{.6}}{\overline{P}^{.2}}\right)$$

$$\text{Class 3: } Duration = 3.181 \times \left(\frac{KWOM^{.6}}{\overline{P}^{.2}}\right)$$

$$\text{Class 4: } Duration = 1.751 \times \left(\frac{KWOM^{.6}}{\overline{P}^{.2}}\right)$$

Table 24. Size Submodel with Size-based Duration Model

Model Without Intercept

Overall Results

SAE = 461 $\overline{MRE} = .55$

Submodel Equations

Class 0:

$$Duration = 2.044 \times KWOM^{.1}$$

Class 1:

$$Duration = 13.619 \times KWOM^{.1}$$

Class 2:

$$Duration = 20.947 \times KWOM^{.1}$$

Class 3:

$$Duration = 18.321 \times KWOM^{.1}$$

Class 4:

$$Duration = 10.424 \times KWOM^{.2}$$

Model With Intercept

Overall Results

SAE = 439 $\overline{MRE} = .55$

Submodel Equations

Class 0:

$$Duration = -147.446 + 139.323 \times KWOM^{.1}$$

Class 1:

$$Duration = 105.470 - 64.434 \times KWOM^{.1}$$

Class 2:

$$Duration = 49.571 - 13.027 \times KWOM^{.1}$$

Class 3:

$$Duration = 60.594 - 21.159 \times KWOM^{.1}$$

Class 4:

$$Duration = 25.003 + 0.065 \times KWOM^{.8}$$

Table 25. Size Submodel with Effort-based Duration Model

Model Without Intercept

Overall Results

SAE = 473 $\overline{MRE} = .79$

Submodel Equations

Class 0:

$$Duration = 4.500 \times MM^{.4}$$

Class 1:

$$Duration = 11.551 \times MM^{.1}$$

Class 2:

$$Duration = 17.837 \times MM^{.1}$$

Class 3:

$$Duration = 16.581 \times MM^{.1}$$

Class 4:

$$Duration = 3.968 \times MM^{.3}$$

Model With Intercept

Overall Results

SAE = 412 $\overline{MRE} = .64$

Submodel Equations

Class 0:

$$Duration = -10.530 + 19.949 \times MM^{.1}$$

Class 1:

$$Duration = 97.787 - 50.490 \times MM^{.1}$$

Class 2:

$$Duration = -5.335 + 20.929 \times MM^{.1}$$

Class 3:

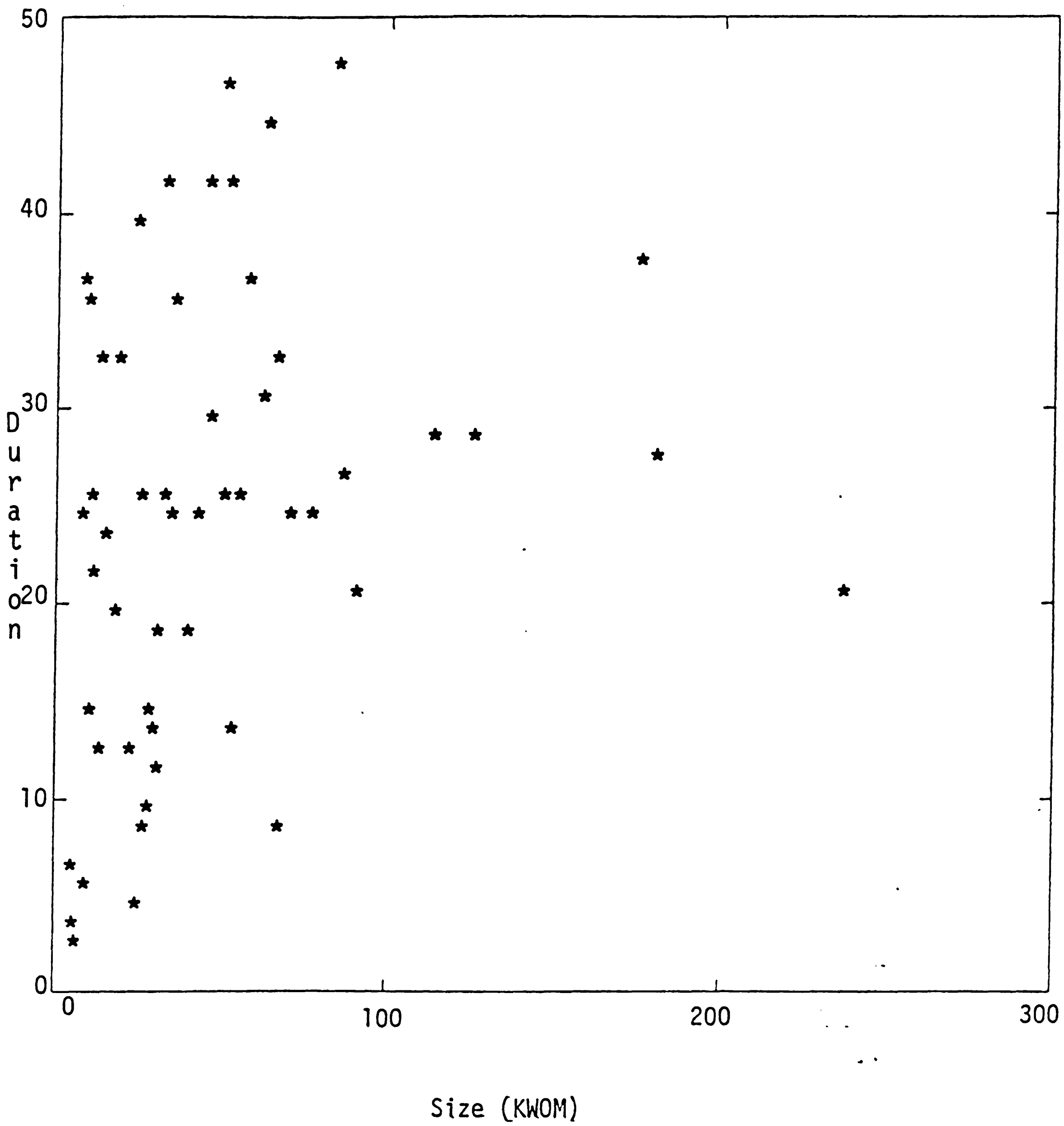
$$Duration = 25.878 + 1.334 \times MM^{.1}$$

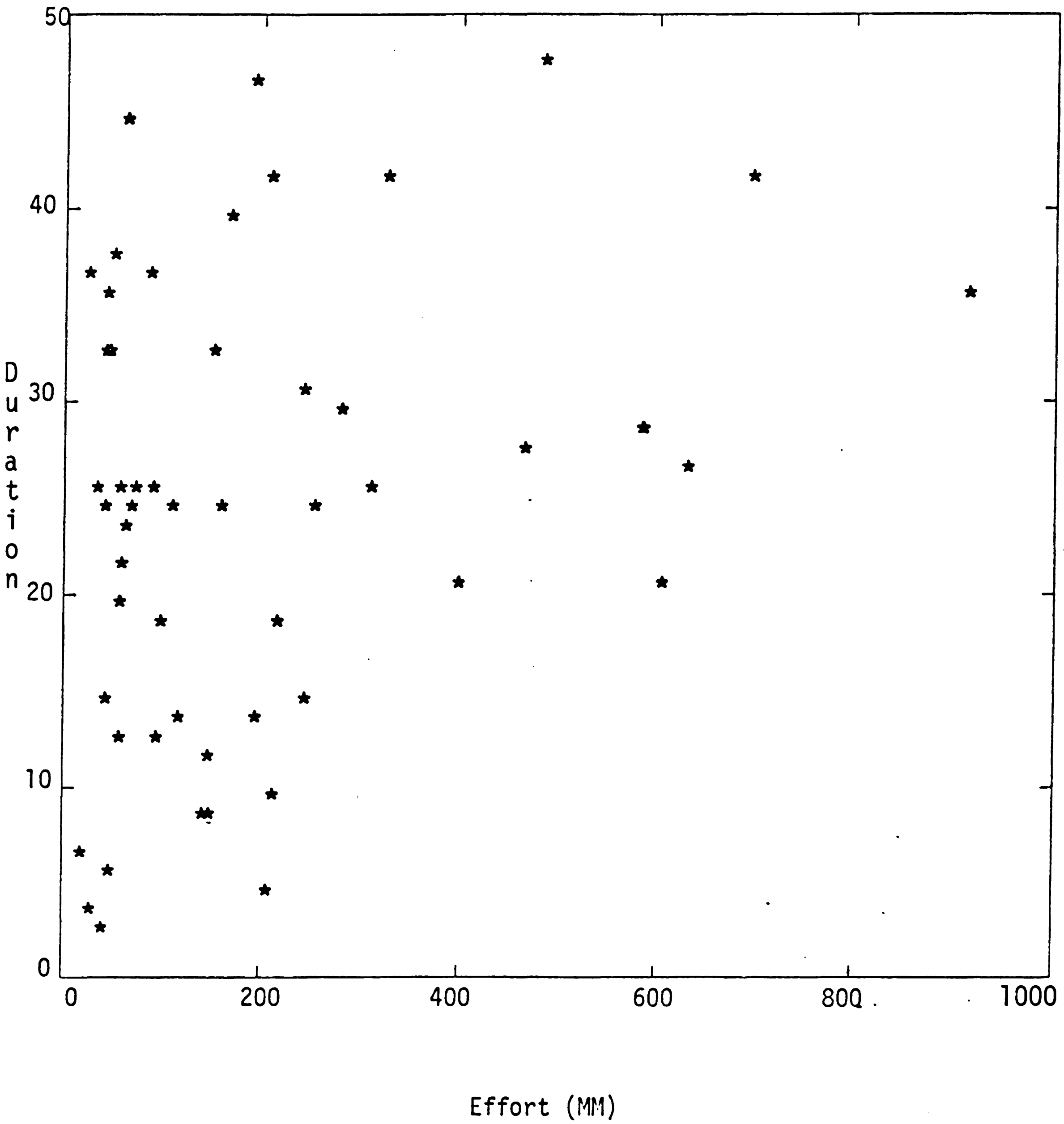
Class 4:

$$Duration = 7.974 + 0.001 \times MM^{1.1}$$

The results of the size submodelling on the duration equations are an improvement over the single duration models. However, the performance of the duration submodels is not at an acceptable level. The SAE values are all above 400 and the $\overline{MRE}$ values are all above .50. The accuracy needed is similar to that needed in effort prediction, i.e. $\overline{MRE}$ values less than .20.

The failure of the duration submodelling exhausts available alternatives for modelling duration. Finding reasons why the data is difficult to model, is the next direction of work. Examining the data shows the primary difficulty in the duration data. The following graphs represent size versus duration, and effort versus duration respectively. It can be seen that no linear relation exists between either of the pairs of product attributes. The primary reason for the nonlinearity is that duration has a very small variance in comparison to size or effort. Also, duration often acts independently of size due to varying complexity and other factors that make two similar sized products have radically different durations.


50
40
30
20
10
0
Duration
0
100
200
300
Size (KWOM)


50
40
30
20
10
0
Duration
0
200
400
600
800
1000
Effort (MM)

Risk

Another approach attempting to locate ways to verify effort estimates is to determine the risk of predictions. The goal of the work in the area of risk is to determine the effects of inaccuracy in the values used to generate effort estimates. The work should yield the sensitivity of the model to error in size and $\overline{P}$. Table 26 is an example of the effects of error in size and $\overline{P}$ on effort. The example was created using a positive and negative ten percent error in size and $\overline{P}$ in the single model COPMO effort predicting equation.

The primary result shown in Table 26 is that the model is more sensitive to error in the $\overline{P}$ than it is to size. This is demonstrated by moving horizontally across the table and seeing only one MM change, whereas moving vertically down the table there is a twelve MM change in the values. This situation is not suprising considering the format of the COPMO equation. Given that size is only modified by a multiplier, and that $\overline{P}$ is modified by a multiplier and an exponent, it is expected that $\overline{P}$ will be the more sensitive variable.

Table 26. Affects of Error in Size and P

	Size-10%	Size	Size + 10%
$\overline{P}$-10%	112.37	113.11	113.85
$\overline{P}$	123.51	124.25	124.99
$\overline{P}$ + 10%	135.43	136.18	136.92

Chapter 7

Conclusions

Within the realm of software estimating models, this study has accomplished a variety of valuable tasks. The primary accomplishment is the testing of the relatively new COPMO model by Thebaut in a new environment. Although the implementation used in this study was a slight variation on Thebaut's method, the positive results achieved with the COPMO model demonstrate its value. A summarization of the COPMO results, reveal that it consistently performs as well as, and usually better than the other models. The superiority of the COPMO model is shown by a consistently lower $\overline{Error}$ and $\overline{RE}$. The best product level model is the COPMO duration submodels. The best project level model is the constructive modelling technique used on the COPMO duration submodels. Both the best project model and best product model have the $\overline{MRE}$ within twenty percent which is considered acceptable according to the current modelling literature [Dem82].

The Basic COCOMO model by Boehm shows varied performance throughout the study. Boehm's model occasionally matches the performance of the COPMO model, but more often the performance is mediocre. Considering the simplicity of the Basic COCOMO model, the results achieved are fairly commendable. The occassional good results of the COCOMO model show that the primary driver of required effort is the size of the product. Size is a driver since it is the most measureable attribute of a product and there is inherent relationship between given quantums of effort and size. The COCOMO model shows its best results in the language submodels, the product submodels and the duration submodels. For all three submodels the $\overline{MRE}$ is in the 60 to 70 percent range which is not acceptable but is still considrably better than the Putnam model. The other characteristic behavior of the COCOMO model is its $\overline{RE}$ value being predominantly close to zero. This characteristic infers that the model tends to equally overestimate and underestimate. Although the characteristic may seem insignificant, it could be highly advantageous when used to form constructive project estimates since often the overestimates would cancel underestimates.

The Putnam model has received public acceptance and praise, however, the acclaim was centered around the primary equation, the equation which predicts the needed effort at a given point in time during the project life.

$$Y(t) = K/D^2te^{-t^2/2D^2} \qquad \text{(from chapter 1)}$$

The equation used in the study,

$$Effort = c \times \frac{KSLOC^3}{Duration^4} \qquad \text{(from chapter 1)}$$

predicts total effort and is derived from the primary equation, but this does not guarantee a similar predicting ability. The results of this study show essentially the opposite. The possible justification for this result is that a successful equation using multiple independent variables is successful due to some inherent correlation between the independent variables. If the equation is converted to yield some other result the inherent correlation between independent variables is gone and so is the success of the equation.

Overall, one must conclude that effort estimate modelling has a long way to go. The best result this study achieved is a thirteen percent $\overline{Error}$. Since the best value is an average, this means there are a number of cases well into the twenty percent error range. If one considers the accuracy of various estimating models in other types of activity it seems reasonable that the same accuracy can be achieved in software effort estimating models. The key to the improvement of effort estimating models is time. Software development is in its adolescent phase and thus still stabilizing. As development stabilizes models will more accurately represent historical databases which will show consistent behavior rather than groups of behavior. The other key to improved modelling is the foundation of models due to the efforts of the 70's. From the wealth of early models, the next generation of models have a far more educated starting point.

Further Studies

Although this study has accomplished several tasks, there is obviously room for further study. The primary area that should be studied using the IBM FSD database is prediction of duration based on size and/or effort. There are a number of models which predict effort (some are listed in chapter 1). Some preliminary work was done using a variety of predicting approaches, however none of the attempts generated an accurate model.

Another direction which should be explored is using other available models. The three models used in this study perform well in representing the spectrum of models. However, every model has a somewhat slightly different flavor, and thus could produce different results. The models which are readily accessible are primarily from the 70's. The models which have been developed in recent could be significantly more accurate and should be attempted.

The other readily available area of study is the implementation and refinement of the model. The data used to produce the equations represents projects which started as much as fifteen years ago. The practices and personnel in software development have inherently improved and thus the current software development behavior is slightly different from that seen in the IBM FSD database. Therefore when the COPMO equations are used initially, this situation must be at least considered.

As the first projects which used the COPMO model to estimate effort are completed, the equations should be recalibrated to check the equation's constants.

Appendix A. Adjustment Multipliers

The following tables are the multiplier values for the estimates generated for the product submodels, the size submodels and the duration submodels. The method to use these values was explained in chapter 4.

Table 27. Language Adjustment Multipliers for Product Submodel Estimates

Language HO
  Effort = 0.968 * Product Effort Estimate
Language SL
  Effort = 1.109 * Product Effort Estimate
Language AS
  Effort = 1.001 * Product Effort Estimate

Table 29. Duration Adjustment Multipliers for Product Submodel Estimates

Class 0: (KWOM < 10)
Effort = 0.626 * Product Effort Estimate
Class 1: (10 < = KWOM < 30)
Effort = 0.770 * Product Effort Estimate
Class 2: (30 < = KWOM < 60)
Effort = 0.960 * Product Effort Estimate
Class 3: (60 < = KWOM < 100)
Effort = 1.010 * Product Effort Estimate
Class 4: (KWOM > = 100)
Effort = 1.000 * Product Effort Estimate

Table 30. Language Adjustment Multipliers for Size Submodel Estimates

Language HO
Effort = 1.004 * Size Effort Estimate
Language SL
Effort = 1.097 * Size Effort Estimate
Language AS
Effort = 0.998 * Size Effort Estimate

Table 31. Product Adjustment Multipliers for Size Submodel Estimates

Type 1:
Effort = 0.997 * Size Effort Estimate
Type 2:
Effort = 0.999 * Size Effort Estimate
Type 3:
Effort = 1.155 * Size Effort Estimate
Type 4:
Effort = 1.030 * Size Effort Estimate
Type 5:
Effort = 1.158 * Size Effort Estimate

Table 28. Size Adjustment Multipliers for Product Submodel Estimates

Class 0: (Elapse < 10)
  Effort = 0.430 * Product Effort Estimate
Class 1: (10 < = Elapse < 20)
  Effort = 0.739 * Product Effort Estimate
Class 2: (20 < = Elapse < 30)
  Effort = 1.132 * Product Effort Estimate
Class 3: (30 < = Elapse < 40)
  Effort = 1.005 * Product Effort Estimate
Class 4: (Elapse > = 40)
  Effort = 0.998 * Product Effort Estimate

Table 32. Duration Adjustment Multipliers for Size Submodel Estimates

Class 0: (Elapse < 10)
  Effort = 0.887 * Size Effort Estimate
Class 1: (10 < = Elapse < 20)
  Effort = 0.670 * Size Effort Estimate
Class 2: (20 < = Elapse < 30)
  Effort = 0.930 * Size Effort Estimate
Class 3: (30 < = Elapse < 40)
  Effort = 1.009 * Size Effort Estimate
Class 4: (Elapse > = 40)
  Effort = 1.006 * Size Effort Estimate

Table 33. Language Adjustment Multipliers for Duration Submodel Estimates

Language HO
  Effort = 1.006 * Duration Effort Estimate
Language SL
  Effort = 1.020 * Duration Effort Estimate
Language AS
  Effort = 0.999 * Duration Effort Estimate

Table 34. Product Adjustment Multipliers for Duration Submodel Estimates

Type 1:
Effort = 0.990 * Duration Effort Estimate
Type 2:
Effort = 1.035 * Duration Effort Estimate
Type 3:
Effort = 0.999 * Duration Effort Estimate
Type 4:
Effort = 1.045 * Duration Effort Estimate
Type 5:
Effort = 0.997 * Duration Effort Estimate

Table 35. Size Adjustment Multipliers for Duration Submodel Estimates

Class 0: (KWOM < 10)
Effort = 0.689 * Duration Effort Estimate
Class 1: (10 < = KWOM < 30)
Effort = 1.032 * Duration Effort Estimate
Class 2: (30 < = KWOM < 60)
Effort = 1.003 * Duration Effort Estimate
Class 3: (60 < = KWOM < 100)
Effort = 0.962 * Duration Effort Estimate
Class 4: (KWOM > = 100)
Effort = 1.001 * Duration Effort Estimate

Bibliography

[Aro77] Aron, J. D., "Estimating Resources For Large Programming Systems," *COMPSAC77 Tutorial*, IEEE, November 1977, pp. 257-268.

[Bau79] Bauer, T. H., "Software Cost Estimating Experience," *IEEE Workshop on Quantitative Software Models*, IEEE, October 1979, pp. 78-81.

[Bas78] Basili, V. R. and Zelkowitz, M. V., "Analyzing Medium- Scale Software Development," *Proceedings, 3rd International Conference On Software Engineering*, 1978, pp. 116-123.

[Bas80] Basili, V. R., "Resource Models," *Tutorial on Models and Metrics For Software Management and Engineering*, IEEE, 1980, pp. 4-9.

[Boe78] Boehm, B. W. and Wolverton R. W., "Software Cost Modeling: Some Lessons Learned," *Second Software Life Cycle Management Workshop Proceedings*, U.S. Army CSC and IEEE, August 1978, pp. 129-132.

[Boe81] Boehm, B. W., *Software Engineering Economics*, Prentice Hall Inc., Englewoods Cliffs, NJ., 1981.

[Boy84] Boydston, R. E., "Programming Cost Estimate: Is It Reasonable?", *7th Int. Conference On Software Engineering*, ACM, IEEE and National Bureau of Standards, 1984, pp. 153-159.

[Bro75] Brooks, F. P., *The Mythical Man-Month*, Addison-Wesley Inc., Reading, MA, 1975

[Bru82] Bruce, P. and Pederson, S. W., *The Software Development Project - Planning and Management*, John Wiley and Sons Publishing Inc., New York, NY., 1982.

[Bry83] Bryant, A. and Kirkham, J. A., "B. W. Boehm Software Engineering Economics: A Review Essay," *ACM SIGSOFT Notes*, Vol. 8, No. 3, July 1983, pp. 44-59.

[Buc77] Buckle, J. K., *Managing Software Projects*, American Elsevier Publishing Company Inc., New York, NY., 1977.

[Cru82] Cruickshank, R. D., and Lesser, M., "An Approach to Estimating and Controlling Software Development Costs," in R. Goldberg (ed.), *The Economics of Information Processing, Volume 2*, John Wiley & Sons, Inc., 1982, pp. 139-148.

[Cru85] Cruickshank, R. D., personal correspondence, September 1985.

[Dem82] DeMarco T., *Controlling Software Projects*, Yourdon Press, New York, NY., 1982.

[Eva83] Evans, M. W., Piazza P. and Dolkas J. B., *Principles of Productive Software Management*, John Wiley and Sons Inc. New York, NY., 1983.York, NY., 1983.

[Fox82] Fox, J. M., *Software and Its Development*, Prentice-Hall Inc., Englewood Cliffs NJ., 1982.

[Fre79] Freiman, F. R. and Park, R. E., "The Price Software Cost Model," *NAECON Proceedings*, IEEE, 1979, pp. 280-288.

[Gaf79] Gaffney, J. E. Jr and Heller, G. L., "Macro Variable Software Models For Application To Improved Software Development Management," *IEEE Workshop On Quantitative Models*, IEEE, October 1979, pp. 63-68.

[Hal77] Halstead, M. H., *Elements of Software Science*, Elsevier North-Holland, Inc., New York, NY, 1977.

[Han82] Hannan, J. (editor) and Oliver, P., *A Practical Guide To Computer Programming Management*, Auerbach Publishers Inc., Pennsauken, NJ., 1982.

[Hor75] Horowitz, E. (editor) and Wolverton, R. W., *Practical Strategies For Developing Large Software Systems*, Addison-Wesley Publishing Company, 1975.

[Jun79] Junk, W. S., "A Software Cost Estimation Methodology Creating The Structure For Reliable Application Of State Of The Art Cost Models'," *IEEE Workshop On Quantitative Models Proceedings*, IEEE, October 1979, pp. 56-62.

[Las79] Lasher, W., "Software Cost Evaluation and Estimation: A Government Source Selection Case Study," *IEEE Workshop on Quantitative Models*, IEEE, October 1979, pp. 42-55.

[Nor77] Norden P. V., "Useful Tools for Project Management," *COMPSAC77 Tutorial*, IEEE, November 1977, pp. 113-143.

[Par80] Parr, F. N., "An Alternative to the Rayleigh Curve Model for Software Development Effort," *IEEE Transactions on Software Engineering*, IEEE, May 1980, pp. 291-296.

[Put77] Putnam, L. H., "The Software Life Cycle: Practical Application to Estimating Cost, Schedule and Providing Life Cycle Control," *Compsac77 Tutorial*, IEEE November 1977, pp. 39-112.

[Rub83] Rubin, H. A., "Macro-Estimation of Software Development Parameters: The ESTIMACS System," *SoftFair Proceedings*, IEEE, National Bureau of Standards, and ACM, July 1983, pp. 109-118.

[San79] Sandler, G. H. and Rachowitz, B. I., "Software Cost Models - Grumman Experience," *IEEE Workshop on Quantitative Models*, October 1979, pp. 69-77.

[Sho79] Shooman, M. L., "Tutorial On Software Cost Models," *IEEE Workshop On Quantitative Software Models*, IEEE, 1979, pp. 1-19.

[Sho83] Shooman, M. L., *Software Engineering*, McGraw-Hill Book Company, New York, NY, 1983.

[Ste77] Stephenson W. E., "An Analysis of the Resources Used in the Safeguard System Software Development," *COMPSAC77 Tutorial*, IEEE, November 1977, pp. 303-312.

[The83] Thebaut, S. M., "The Saturation Effect In Large-Scale Software Development: Its Impact and Control," *Computer Sciences Department Division of Mathematical Sciences Purdue University*, 1983.

[Tur84] Turner, R., *Software Engineering Methodology*, Reston Publishing Company Inc., Reston, VA., 1984.

[Wal77] Walston, C. E. and Felix C. P., "A Method of Programming Measurement and Estimation," *IBM Systems Journal*, Vol. 16, No. 1, 1977, pp. 54-73.

[Wol77] Wolverton, R. W., "Quantitative Management: Software Cost Estimating," *COMPSAC77 Tutorial*, IEEE, November 1977, pp. 145-255.

The vita has been removed from the scanned document