SOFTWARE DEVELOPMENT COST ESTIMATING MODELS- AN APPLICATION OF THE HEDONIC PRICING APPROACH

by

Kirsten M. Pehrsson

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 ARTS
in
Economics

APPROVED:

Dr. David Meiselman

Dr. Alan Freiden

Dr. Robert Mackay

June, 1987

Blacksburg, Virginia

SOFTWARE DEVELOPMENT COST ESTIMATING MODELS- AN APPLICATION OF THE HEDONIC PRICING APPROACH

by

Kirsten M. Pehrsson

Committee Chairman: Dr. David Meiselman
Economics

(ABSTRACT)

Software development cost estimating models were analyzed using an application of the hedonic pricing approach. Several recently proposed software cost estimating tools were surveyed for the purpose of revealing their roots in the hedonic pricing approach. The analysis includes discussion of the hedonic pricing approach, the logic of several software cost models, and anlysis of the models' hedonic pricing traits.

Hedonic prices are the implicit prices of attributes of differentiated products as revealed in the market through attribute levels associated with market-clearing prices. Several aspects of the software costing models fit within the hedonic pricing approach. Many of the models base cost estimates on the varying quantities of software product attributes (e.g., complexity of program, schedule requirements, etc.). Similarities and differences of traits among cost models were noted.

TABLE OF CONTENTS

SOFTWARE DEVELOPMENT COST ESTIMATING MODELS- AN APPLICATION OF THE HEDONIC PRICING APPROACH

I. INTRODUCTION

This paper examines software development cost-estimation models as forms of hedonic price regression functions. Through analysis of existing estimation models, it will be determined whether or not these models exhibit characteristics of hedonic pricing relationships. This analysis will involve exploring the hedonic pricing model, as presented by Rosen, and its similarities with the software cost models. It will also involve examination of whether the models are based upon pricing relationships which are similar, vastly different, or somewhere in-between. The study will conclude with an exploration of what the future might hold for attempts to derive hedonic pricing relationships in the software development field.

The motivation for studying this particular topic is four-fold. Firstly, being employed in the cost-estimating field, one becomes interested in this fairly new and special area of cost-estimating models. Personal experience in developing data bases and simple estimating equations for hardware acquisitions for over two years has not involved exposure to any software cost-estimating methodology. Secondly, software development is an ever-increasing sector of expenditures, both in government (especially the

military) and in the private sector. This makes the accuracy and reliability of the software cost-estimating models essential for making efficient investment decisions. Thirdly, the particular situation of estimating production functions for a heretofore undeveloped field presents factors which add a new twist to a traditional economics problem. Finally, by examining the evolution of the software cost-estimating models, a definition of some trends in the software cost-estimating art/science, and a projection of directions to be taken by future models will be attempted.

II. HEDONIC PRICING AND SOFTWARE COSTS MODELS

Rosen describes hedonic prices of goods as "the implicit prices of attributes and are revealed to economic agents from observed prices of differentiated products and the specific amounts of characteristics associated with them." [1] In other words, the hedonic prices are the first-order coefficients of the attributes in a regression of product prices against the attributes that they contain. Rosen examines hedonic prices as a problem in the economics of spatial equilibrium in which the entire set of implicit prices guides both consumer and producer locational decisions in characteristics space.

Rosen examines a situation of goods containing n characteristics described by z, where $z = (z_1, z_2, \ldots, z_n)$. A price, $p(z) = p(z_1, z_2, \ldots z_n)$ is defined at each point on the vector of attribute combinations. The price determines both consumer and producer choices of characteristics of goods transacted. Rosen proceeds to show how to determine parameters in a hedonic price regression equation by studying the prices of varying models, brands, etc. of a class of commodities. The consumption decision is analyzed through the individual's family of indifference surfaces relating amounts of zi with money foregone to obtain those

---

[1] Sherwin Rosen, "Hedonic Prices and Implicit Markets: Product Differentiation in Pure Competition," Journal of Political Economy, January/February 1974, p.3355.

given a constant utility index and income. A similar analysis is performed for the producers, but where indifference surfaces relate z_i with minimum unit prices the firm is willing to accept given constant profits. Equilibrium is obtained when the consumer's value and producer's offer function are tangent, with the point of tangency defining the market-clearing price function, $p(z)$. Buyers have different taste preferences, and firms have differing factor costs, which creates a range of value and offer functions. Because the offer and value functions vary between sellers and buyers there is range of "offer-value tangencies", giving a range of market-clearing prices for different values of z_i.

The consumption decision as discussed by Rosen can be portrayed graphically (see Fig. 1.) The θ curves represent the amount the consumer is willing to pay for z at a fixed utility index and income. The p curve represents the minimum price demanded on the market. Utility is maximized at the tangency of the consumer's curve with the market curve. The difference in tastes of the two consumers is shown by the differing points of tangency -- one is willing to pay more for greater amounts of the particular characteristic z_1 than the other (when the components of z $(z_2, \ldots z_n)$ are held constant.

The production decision is, in effect, a "mirror image"

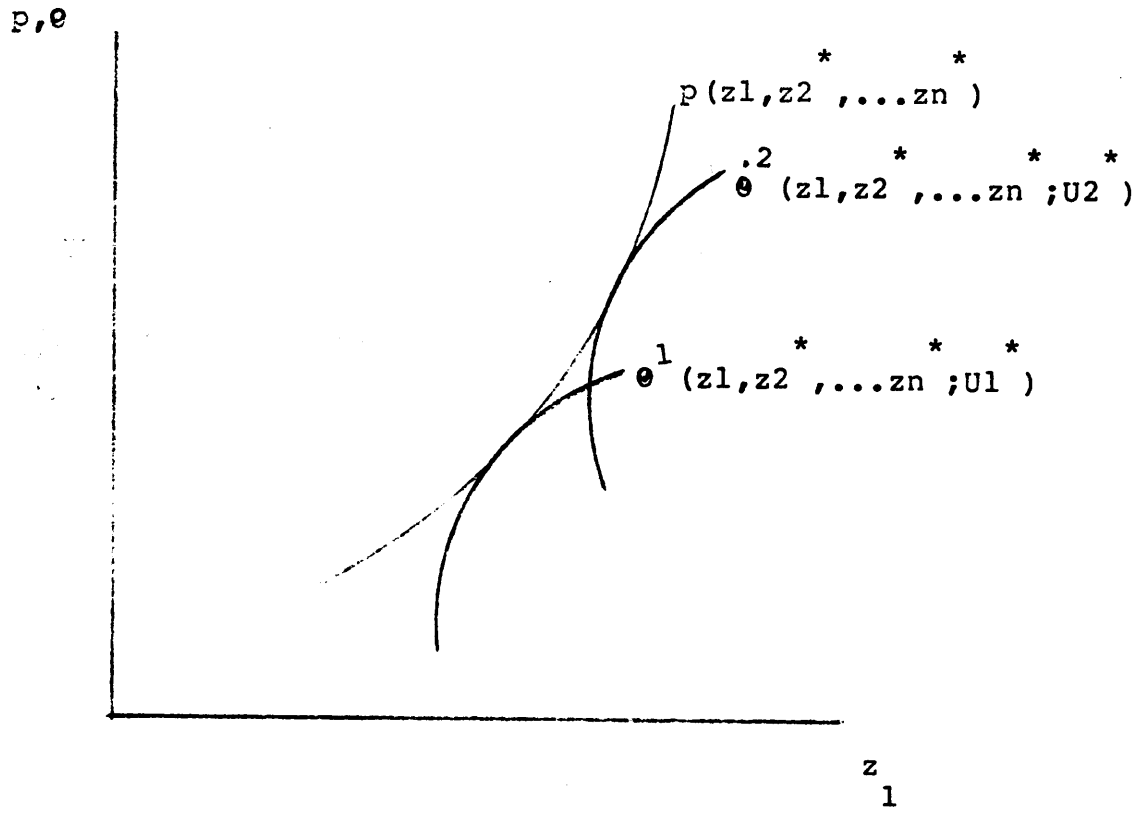


FIGURE 1
CONSUMPTION DECISION IN ROSEN MODEL

of the consumption decision. Here, rather than differing consumer taste preferences, there is the effect of differing marginal costs of attributes (see Fig. 2.) This can be a result of factor prices or "technology" (typically unmeasured, firm-specific factor of production.) ϕ represents the marginal reservation supply price for z_1 assuming constant profit and fixed $z_2^* \ldots z_n^*$ (where $z_2^* \ldots z_n^*$ are at optimum values.) Obviously, one firm is more well-suited to produce less amounts of z_1, and the other more. The equilibrium (where the curves are tangent) represents where the offered price can actually be demanded on the market. Consequently, a variety of "packages" of characteristics appear on the market to satisfy differences in consumer preferences. These distributions of consumer tastes and producer prices determine market price.

In the simple example given by Rosen, there is only one characteristic, or quality measure, z_1. The consumer will purchase where the MRS (ϕ) is equal to dp/dz_1 i.e., the point of tangency with the offer curve. The value functions in this case are straight lines. The hedonic price can be shown by the solution to "Euler's equation," which is of form $p = c_1 z_1^r + c_2 z_1^s$ where c_1 and c_2 are constants determined by the boundary condition. The situation can be depicted graphically (see Fig. 3.) Depending on the boundary conditions, the hedonic price is shown in the region between $z_{1(0)}$ and $z_{1(2)}$. Firms will not produce at

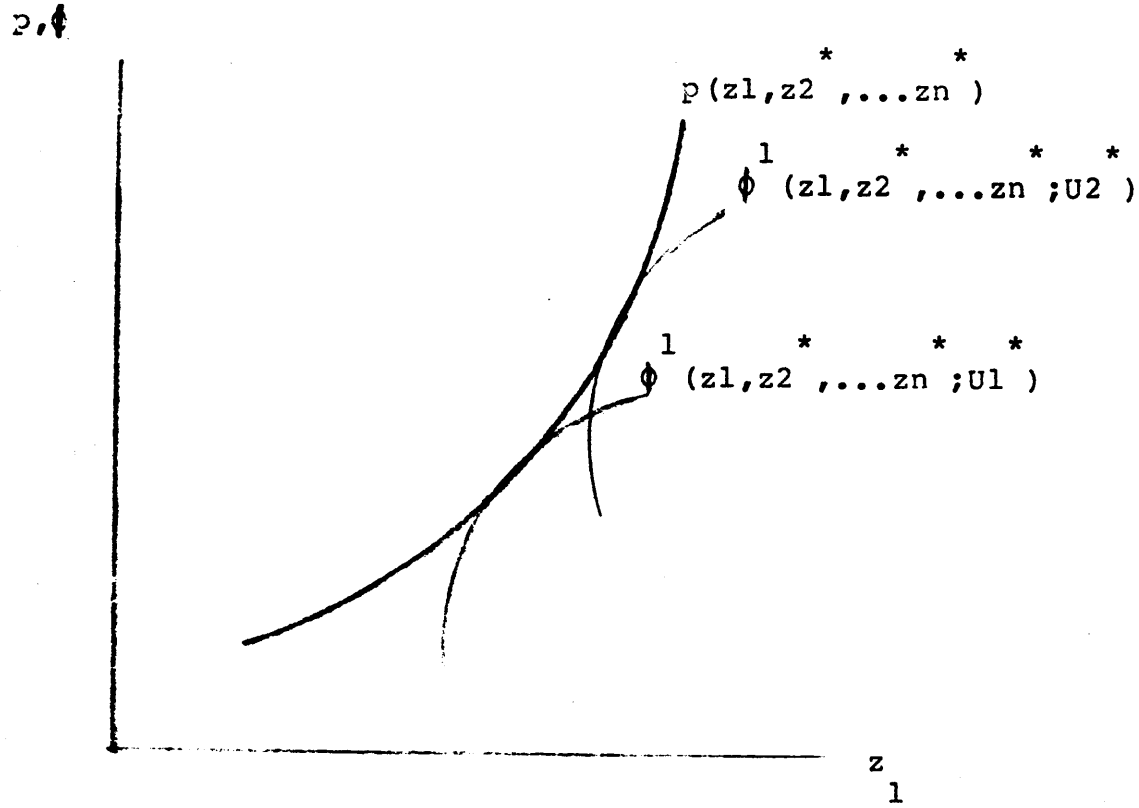


FIGURE 2
PRODUCTION DECISION IN ROSEN MODEL

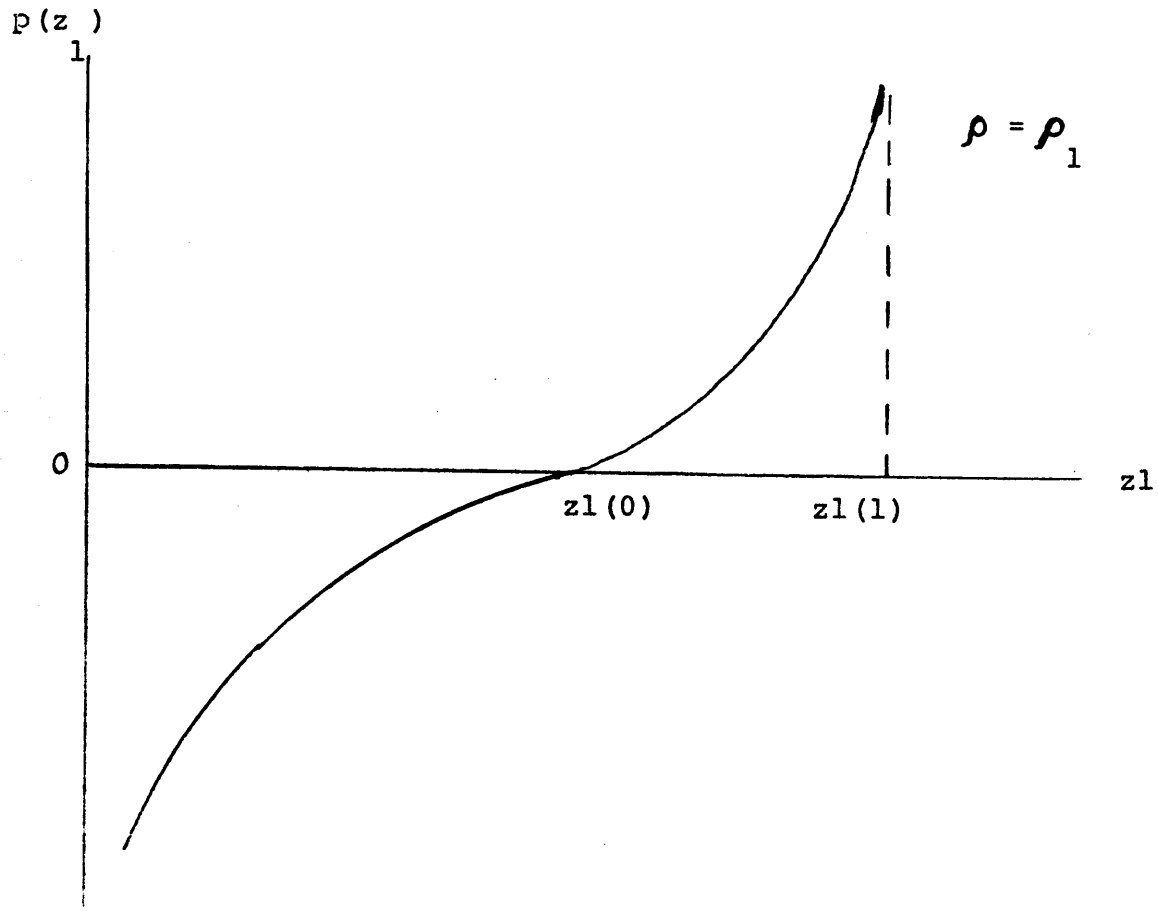


FIGURE 3
GRAPHICAL DEPICTION OF HEDONIC PRICE EFFECT

less than $z_{1(0)}$ as they choose not to sell at negative prices.

Rosen's criticism of the typical derivation of marginal hedonic prices is that they merely connect equilibrium reservation prices and characteristics, and reveal little about underlying supply and demand functions. He suggests an alternative method of estimation in which the variable accounting for differences in consumers and producers is considered. The method involves a two step procedure. The first step is to estimate the p(z) by the usual regression method, without considering the effect of individual consumer or producer preferences. He notes that this duplicates the information acquired by observers in the market. Then, one must compute a set of implicit marginal prices for each buyer and seller, evaluated at the amounts of characteristics (attributes) bought/sold. Then, one uses the estimated marginal prices as endogenous variables to solve for the hedonic equations incorporating effects of individual preferences.

Rosen uses the analytic structure presented to show the welfare consequences of quality-standards legislation. The example used is the imposition of a law to force consumers to purchase a good containing a minimum of some attribute. It shows that consumers will be worse-off than if they would have purchased less than the minimum amount before the law was effective.

Some of the attributes of hedonic price relationships described in the Rosen article are apparent in the software cost estimating models. Software cost estimating models are, in effect, an attempt to derive the hedonic price relationship underlying the market for software development. The following sections will illustrate that most of the existing models are based upon simple regression-type equations. The equations estimate the dependent variable, in this case the effort to develop the software system, based upon one or more independant variables. The independant variables are the factors in the model deemed to affect development effort. The general approach used to derive these relationships is similar to that described to derive hedonic prices in Rosen's article. Rosen describes observation of hedonic prices through the market prices associated with different brands and models (and thus different amounts of attributes) for a particular class of goods.

Although software is not a "tangible" good, (in fact it could be thought of more as a service) the same approach is used. Software does not come in different brands (as one might think of brands of detergent) per se. However, it does come in varying levels of certain attributes. Examples of software attributes are: complexity of application, level of language, user-friendliness, response-time, and many more. Some of these attributes are more important to the overall

development effort of the project than are others. Defining which are the important attributes is equivalent to declaring which "z_i"s are included in the regression equation. Estimating the parameters of these variables is another challenge to cost-estimators. Both of these procedures are sources of controversy, as will be illustrated in the following examination of the models available today.

III. SPECIAL FACTORS IN COSTS OF PRODUCING SOFTWARE

Software costs are eating-up an increasingly large share of private industry and government budgets. The trend is particularly apparent in the U.S. military expenditures. Until the last twenty years, hardware has accounted for an overwhelming majority of defense system procurement expenditures. Recently, however, has proven to be a different story. In this new age of electronic sophistication and automation, weapon systems are becoming increasingly dependent upon the integration of computers into their operation. A recent projection for the Department of Defense software budget for 1990 exceeds $32 billion. Software is a dominant part of the information processing industry which is expected to achieve 13% of the GNP by 1990.[2] To use a timely illustration, a recent prediction estimated that the Strategic Defense Initiative (SDI) program will require about 10 million lines of code![3] Consequently, software development and software maintenance are absorbing a larger share of the defense dollar than ever before. The traditional "guns and butter" economic analysis might today be more aptly termed "code and

---

[2] Randall Jenkins and Suzanne Lucas, "Sensitivity Analysis of the Jensen Software Model." ISPA Conference Proceedings, p. 384.

[3] Dr. Robert Setzer, "Spacecraft Software Cost Estimation: Striving for Excellence Through Parametric Models (A Review)," ISPA Conference Proceedings, 1986,p.600.

butter."

This trend of weapons sophistication and software integration presents problems for those responsible for estimating weapon and defense system acquisition costs. Traditional cost-estimating techniques are tailored towards typical hardware (guns, tanks, etc.) acquisitions. The techniques are based on parameters which are not applicable to a software program. For example, such values as weight, number of components, and power source, while useful parameters for a radio or jammer, are of little relevance to a computer program. Meanwhile, the parameters which <u>are</u> relevant for software acquisiton costs are not yet clear. Merit of the sometime-used "lines-of-code" as a sufficient parameter has been debated, for example, as it does not account for impacts of human productivity, experience with similar systems, and other factors which could affect programming costs.

There is another complicating factor in the software cost-estimating dilemma. Many of the hardware cost-estimating methodologies are parametric models based on data collected on past procurement of systems. Because the procurement cycle may last for several years, this presents a special problem for software cost estimators. As "software-heavy" system procurements are relatively recent, there is not a large amount of historical and complete cost data from which to build a model. Regardless, because of

the increasing importance of software costs, it is essential to develop some type of methodology. A result has been the appearance of several models in the past few years which are tailored particularly for estimating software costs.

Software cost-estimating has several special considerations. First, the human factor plays a larger role in software development than in a typical manufacturing process. Whereas an assembly line's efficiency is highly dependent on tangible factors such as equipment, available raw materials etc., software development is more highly dependent upon non-tangible resources such as programmer's experience, organizational abilities, and communication abilities. Secondly, in the heat of finishing a project, tracking the development of a software system is often neglected. A good record is not always kept of development specifications, hours spent programming, coding, etc. Careful documentation of a project is a time-consuming and expensive task. Both of these factors add to the difficulty of obtaining extensive, clearly defined data on which to base an estimating model.

IV. SURVEY OF MODELS (HEDONIC RELATIONSHIPS)

There are several software cost estimation techniques that are used, each with their relative strengths and weaknesses. Boehm, considered by many to be the "father" of software cost estimating, outlines several in his book [4]. Estimation by analogy involves planning the cost of a project by looking at the actual costs of similar completed projects. Expert judgement involves consulting with experts in the field of the cost to be estimated. Boehm notes that this technique may be aided by an expert-consensus mechanism such as the Delphi technique. The Parkinson principle that "work expands to fill the available volume" implies that all available resources will be used in a project. A fourth method is Price-to-win, where the cost estimate given is what is considered to win the job (in a bid situation). Top-down estimates use general properties of the software effort to estimate total costs, and then the cost is split up among the components of the project. In bottom-up estimates, on the other hand, the cost of the project components are estimated first, and then aggregated to give a total project cost. Finally, there are the algorithmic models. In this case, software costs are estimated as a function of variables that are considered to be important

---

4Barry Boehm, "Software Engineering Economics," IEEE Transactions on SOftware Engineering, Vol. SE10, No. 1, January 1984. p. 10.

cost drivers. Types of algorithmic methods include: regression, heuristic, and phenomenological. Boehm notes that the Parkinson and price-to-win methods are unacceptable for use in realistic cost estimates. The last method, the algorithmic method, is the most prevalent. It is this method which shall be next analyzed as a form of hedonic price regression.

COCOMO

Model Structure.

Perhaps the most widely-known of the software estimating models was developed by Barry Boehm around 1981. In his book, Software Engineering Economics, Boehm describes a technique called the COnstructive COst MOdel (COCOMO). COCOMO involves three tiers of models which can be used to obtain basic, intermediate, or detailed estimates. The basic model is a single macro-estimation scaling model estimating cost as a function of product size. The detailed model is a micro-estimation model with a three-level work breakdown structure and a set of phase-sensitive multipliers for each cost driver attribute. The intermediate model provides a level of detail between that of the other two models. The last model mode, the intermediate, uses four sets of procedures to determine development costs. These

include: 1) estimating nominal development effort as a function of product's size measured in thousands of delivered source instructions (KDSI) and the project's development mode, 2) determining a set of effort multipliers from the product's rating on a set of 15 cost driver attributes, 3) obtaining estimated development effort by multiplying the nominal effort estimate by the product's effort multipliers, and 4) determination of other specific project elements (dollar costs, development schedules, phase and activity destribution, etc.) from additional factors and the development effort estimate.

The first procedure, nominal effort estimation, involves determining the project's development mode via a table containing descriptive features of the effort. The modes include; 1) organic (more familiar, less constrained projects), 2) semidetached (more constrained, less familiar), and 3) embedded (ambitious, innovative, tightly-constrained projects). The size of the product is then estimated in KDSI. The two factors are then combined in an equation of the form: $(MM)_{nom} = V(KDSI)^{Y}$, where V and Y vary depending upon the mode determined earlier, and MM_{nom} represents nominal effort in man-months.

The second step involves determining the effort multipliers. A table is used to determine what rating (very low to extra high) each of fifteen cost driver attributes should receive. Based upon the rating, a numerical factor

is assigned each of the attributes. This will provide an estimate which accounts for these additional fifteen factors.

There are four groups of factors which Boehm has determined are major cost drivers in software development costs and has included in his estimation process. The first group contains product attributes (required software reliability, data base size, and product complexity). The second contains computer attributes (execution time constraint, main storage contraint, virtual machine volatility, and computer turnaround time). The third contains personnel attributes (analyst capability, applications experience, programmer capability, virtual machine experience, and programming language experience). The fourth group contains project attributes (use of modern programming practices, use of software tools, and required development schedule).

Software estimating models can be considered a derivation of traditional economic production functions. A typical production function involves an entrepreneur utilizing variable input(s) and fixed input(s) to produce a single output, Q. Output (Q) is a function of the variable inputs (X_1, X_2, ...X_n). The production function assumes that the entrepreneur is technically efficient, and that he utilizes the combination of variable inputs to obtain the

highest level of output. In the realm of software estimating models, the output(Q) can be considered to be the positive product attributes (reliability, lines of code, etc.) The typical input of capital could be considered the hardware constraints imposed by available terminals, execution time, etc. for the program development. The typical input of labor could be considered the personnel attributes, such as programming experience of employees, and virtual machine experience. The project attributes, such as required development schedule, and use of modern programming techniques represent separate (mixed labor and capital) variable inputs. Thus, the "cost" equations in the models are a function of the resources required to perform the output, and how much of the output (the product attributes) are required. The estimating models have inherent multi-dimensional production functions using inputs X_1, ... X_n and outputs (software attributes) for Q_1,..n. An exception of the models to the traditional production function is that many of the inputs (such as number of terminals) are not infinitely divisible (completely variable.)

The third procedure is to estimate the development effort. This is achieved by multiplying the nominal development effort by the product of the effort multipliers for the fifteen cost driver attributes. The equation used is of the form $(MM)_{nom} \times (e_1 * e_2 .. e_{15}) = (MM)_{dev}$, where e is

each of the numerical factor for each of the rated cost drivers, and $(MM)_{dev}$ is the estimated development effort (MM = man-months).

The fourth step entails estimating the related project factors. The COCOMO model also provides relationships to compute estimates other than the straight development effort. Equations help to calculate dollar cost of the project, breakdown by effort and life-cycle phase, and type of project activity, project schedule, and phase distribution. The equation for recommended development schedule, for example, is $T_{(dev)} = 2.5*(MM)_{dev}^{X}$, where $T_{(dev)}$ is the recommended months to develop the software, and X is a variable dependant upon the development mode.

An example of the model "in action" is presented by Boehm. A similar example (using different project characteristic assumptions) will be illustrated here. The example involves a scenario of estimating the cost to develop a simple inventory system to track the stock for a neighborhood hardware store. The software producer, dealing solely with developing similar inventory systems, has a high understanding of the requirements, and little innovative programming is needed. Also, manual inventory has been performed for several years, and will continue to be performed concurrent with the operation of new system. Therefore, schedule constraints and product reliability are

not at a premium. The COCOMO model indicates the project should be described as "organic." Then, estimating the size of the product as 5,000 delivered source instructions (5 KDSI), one uses the COCOMO equation prescribed for organic projects: $(MM_{nom} = 3.2\ (KDSI)^{1.05} = 17.34$. One then uses the rating scale and effort multipliers in COCOMO which show how much to account for the project having the rating level for the attribute. The assumed rating and resulting multipliers for the example project are: required software reliability- nominal (1.00), data base size- low (.94), product complexity- low (.85), execution time constraint- (assume all customers call and need to know stock immediately)- high (1.11), main storage constraint- low (0), virtual machine volatility -very low (0), computer turnaround time- nominal (1.0), analyst capability- high (.86), applications experience- high (.91), programmer capability - nominal (1.0), virtual machine experience - low (1.10), programming language experience - low (assume that, although the company has done similar programs, none have been in the language requested by the store owner)- (1.07), use of modern programming practices - high (.91), use of software tools - nominal (1.00), required development schedule- very low (1.23). Note that, logically, those parameters which become more constrictive to the project have a multiplier that is greater than one as they increase and visa-versa. The total "effort multiplier" is the

product of all these (positive) factors, or .914 for this example. This implies an adjustment of 8.6% for the "forgiving" nature of the project. One then calculates the estimated development effort as the nominal effort (17.34 MM, as calculated previously) times the multiplier, .914 = 15.85. The recommended development time is dictated in the equation (for organic projects): $TDEV = 2.5(MM_{dev})^{0.38}$, yielding 7.14 months to develop the simple hardware inventory system in a new programming language. In Boehms example, he estimated an innovative program with estimated 10 KDSI, and the results were an estimated 59 MM of development effort with TDEV = 9 months. The different results show that the effect of size and multiplier on MM is great, but the effect on optimal development time is less.

Because all of the above steps are based upon numerical equations and factors, common sensitivity and tradeoff analyses can be performed. A particular attribute (say, execution time constaint) can be altered, and the equations recalculated to derive a new estimated cost. In this way the effect of changing certain characteristics of the software to be developed can be evaluated in monetary terms. Similarly, the imputed value of the inputs to the firm are known as "shadow" or "accounting" prices in traditional economic analysis.

The COCOMO model obtained the equivalent of a 20%

standard deviation (estimates were within 20% of actuals 68% of the time, with normal distribution of residuals) in a test of 63 sample projects. The developer claims that this is about as good as can be achieved with the state-of-the-art in software estimating techniques, except with a calibration coefficient (a COCOMO option), or in a limited applications context.

Model as Hedonic Pricing Mechanism.

Boehm's model certainly illustrates an attempt to reveal the hedonic pricing (price being measured in man-months) for a software development program. He is seeking to determine which are the main cost drivers, and their respective weights, in a software development program. The relationships he developed were the result of a combination of statistical and logical research in software development costs. In order to evaluate the statistical significance of certain cost drivers, he first had to identify likely candidates. Relating Boehm's effort to Rosen's study of hedonic prices, software development projects would be the class of goods under consideration. Boehm then sought to identify the n characteristics which would determine the development effort in man-months (i.e., the revealed "price") giving the relationship of $p(z) = p(z_1, z_2, \ldots z_n)$, where $p(z)$ is the development effort, and $z_1..z_n$ are the main effort drivers. One difference with Rosen's theoretical model is that in Rosen's model, the goods can be

described by n objectively measured characteristics. Many of the parameters used in Boehm's model are somewhat subjective, in that they are basically ratings given by engineers or managers for the software project. The subjectiveness, actually, is two-fold. First, the model is based upon a data base in which these somewhat subjective parameters were included. Secondly, when the model is actually used, there is more subjectivity in estimating the same parameters for the planned software development project.

The development of COCOMO has an interesting history in that several methods were employed to obtain logical, statistically significant relationships. Boehm himself admits that COCOMO did not employ advanced statistical techniques. He found through preliminary analysis that this approach was not appropriate in the software field for several reasons. First, the software field is too primitive, and cost driver interactions too complex, for the "usual" statistical techniques to be helpful. Secondly, he found that basing functional forms on what was known about software development and life-cycle phenomenology (i.e., logical relationships) was more helpful in achieving usable and believable results.

Boehm's original model was based on 1) a review of existing cost models, 2) a two-round Delphi exercise with 10

software managers to derive effort multipliers (the e variables, mentioned above), and 3) experience with several models at TRW (where Boehm was Chief Engineer of the Software Information Systems Div.). The Delphi exercise involves iterations of anonymous questionnaires to experts in the field, with each iteration hopefully coming closer to a consensus between the experts. This method illustrates the element of expert judgement in the development of the basically algorithmic model. The initial model (which dealt with only one development mode) was calibrated using 12 completed projects and evaluated using 36 projects with satisfactory results. When Boehm expanded his data base of projects to 56, involving more types of development environments, he realized the need to integrate a factor for the development mode. Seven more projects were added to the data base in 1979 and further tests provided good results of accuracy in estimates.

In Boehm's model, as in several of the others, the "price" of the software is revealed in the market, which is defined by the collection of data points of completed projects. The product differentiation of the goods is represented by the different amounts of attributes contained in the projects (level of required reliability, level of storage contraints, etc.)

RCA PRICE-S

Model Structure.

The RCA Price (Price = Programmed Review of Information for Costing and Evaluation) is another highly regarded model. The S represents the software vs. the hardware estimating model. It was developed by Park Frieman around 1979.[5] It consists of using parametric methods to estimate costs and manpower for software development. Frieman defined the main cost driver as software size. The original estimate is refined by several qualitative variables. Among these is a variable called "reso" (resource) which represents the efficiency with which an organization uses its resources to develop a system. [6] This value is obtained by inputing data describing past products within the particular organization, then running the model in the calibration mode. Another variable which varies between users of the model, is called "cplx" (complexity) which describes the familiarity of the staff with the functions to be performed, their general experience, and factors that complicate the development of the system such as new language, more than one user

5Robert Thibodeau, An Evaluation of Software Cost Estimating Models (Huntsville, Alabama, General Research Corporation report prepared for Rome Air Development Center, June 1981, pp. A33 to A62.

6Ibid., pp. A63 to A80.

organization, or state-of-the-art advancement. It reflects the way the organization commits its resources in order to achieve a perceived proper scheduling of a project.

Model as Hedonic Pricing Mechanism

The PRICE-S model lends itself to a comparison with the model in Rosen's article on hedonic pricing. Rosen describes the spectrum of market prices as reflecting a variety of consumer tastes and producer costs with both parties maximizing their behaviour. He elaborates upon the model by examining how both consumers and producers arrive at their positions on the vector of product attributes. In deciding what package of characteristics to assemble, producers evaluate total cost of producing a package as a function of three areas. They are: number of units produced, the specification of characteristics (denoted by z in his analysis), and a variable (defined as B) which represents factor prices and production function parameters which are particular to each seller. An example of factor price differentiation provided is that factor prices may vary for products which are produced in several countries, and traded on national markets. Also, technological differences may shift supply across firms, as when varying levels of education of farm operators may affect their crop yield. Because of this "B" variable, different firms are more suited to produce at different levels of the attributes which are contained in the product. Consequently, producer

equilibrium is characterized by a family of offer functions that are an envelope for the market hedonic price functions. The "B" variable can be considered analogous to the firm-particular variables included in the PRICE-S estimation model. The variable represents an effort to "normalize" the estimating equation across all firms. In light of the analogous relationship between the PRICE-S model and Rosen's study, it would seem to follow that different software producers would be better candidates to produce certain types of software than others. Perhaps one company has a lower "cost per pound"(the variable RESO in PRICE-S), yet has a difficult time adapting to schedule changes. If the project schedule of a program is expected to be volatile (unexpected changes are probable), then perhaps a firm with a higher "cost per pound" of programming would be a better selection for the job. Basically, what Rosen seems to be implying is that a range of market prices for a good is the result of specialization. Unfortunately for a software buyer, such as the military, the actual value of the firm-specific variables are not readily available. Many firms prefer to keep their data proprietary as a strategical aid in the bidding process. In Rosen's model, the market-clearing prices for combinations of good characteristics reveals the efficient producers of particular bundles of characteristics. If one producer is forced to charge more

for a particular bundle than another because of his particular situation (represented by the variable B), the consumer will only buy from the more efficient producer and the first will have to change to producing a different bundle. However, in the software field, the situation is not so simple. Experience as a cost analyst for Dept. of Defense has revealed that contract costs increase drastically as the contractor finds that more funds are needed to complete the project. Similarly, in Research and Development projects, a firm may be under a "cost plus fixed fee" contract, in which they charge according to the expenses incurred. Neither of these cases represents the market situation described by Rosen, where a buyer is able to evaluate exactly what bundle of characteristics, and at exactly what cost, he is going to purchase. In software projects, both the cost and the system characteristics may change drastically between the time the original cost estimate is made, and the time the system is delivered.

SLIM

Model Structure.

A third well-known software estimating model is the SLIM (Software Life Cycle Model) which was developed by L.H. Putnam around 1978. This model is based upon the assumption that the rate of expending effort on the solution of problems follows a Rayleigh distribution function over

time.[6] The basic relationship employed is $S_s = C_k K^{1/3} t_d^{4/3}$ where S_s = number of delivered source instructions, K = life-cycle effort in man-years, t_d = development time in years, and C_k is a technology constant. The variable C_k is calibrated to give a firm-specific representation of the state-of-the-art of technology within the firm. It is affected by such factors as use of modern programming practices, hardware constraints, personnel experience, and interactive development. This variable consequently serves as the "β" used in Rosen's equation, which calibrates the each firm for its specific circumstances, as described in the previous model.

An interesting aspect of SLIM is the relationship which Putnam has determined exists between the project development time and development effort. As the equation demonstrates, he feels that a slight increase in development time (t_d) can cause a great reduction in cost (K). Boehm gives the example that, using this equation, an increase in development time from 10 to 12 months would cause costs to be cut in half.[7] Another interesting aspect of his model is that he has incorporated a linear programming solution which solves for a range of development times and efforts (costs), given constraints on the other variables. The

---

[7] Barry Boehm, "Software Engineering Economics," IEEE Transactions on Software Enigneering, Vol. SE10, No. 1, January 1984. p. 10.

output from this model shows the combinations which are feasible, from the minimum time(to complete)-higher cost solution to the minimum-cost/higher-time solution.

Model as Hedonic Pricing Mechanism.

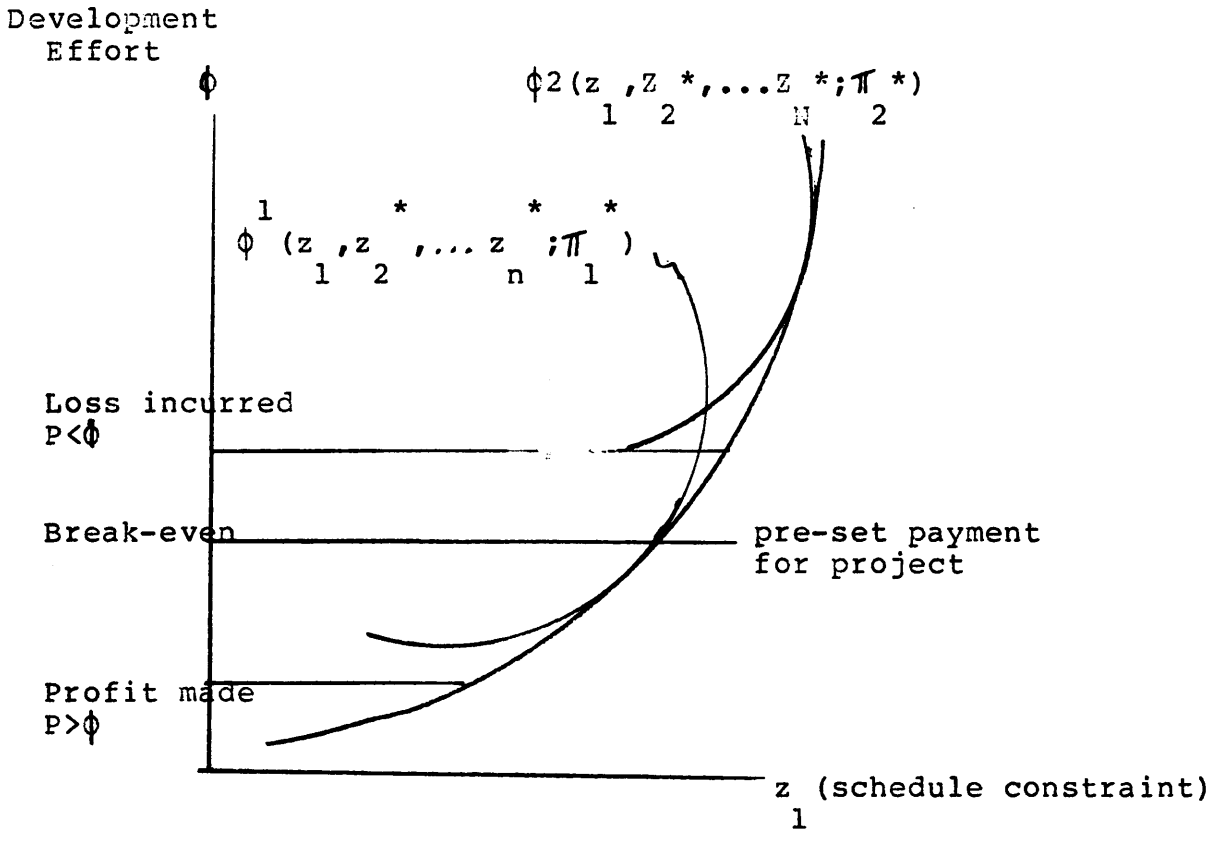
The approach used in SLIM seems to be very similar to the derivation of the profit-characteristics indifference curve for the producer in Rosen's analysis. In Rosen's example, he portrays a graph of the indifference curve between varying levels of one attribute (z_1), and the minimum price the producer is willing to accept for the level of that attribute (at a constant profit rate). This curve represents the producer's offer function, ϕ. In the SLIM model, varying levels of development time(Z_1) with the other variables ($z_2..z_n$) being constrained, are used to calculate what the corresponding development effort would be. Using this information, the producer would be able to determine the most advantageous approach to the project, given the market prices for the different amounts of the attribute z_1.

Another interesting aspect of the SLIM model is that it allows for uncertainty in specifying the independant variables. This means that a range may be specified for estimating a variable such as program size, if accurate estimation is difficult. The output from this use of this mode of the model has direct application to risk analysis of

different approaches to the project. This methodology is equivalent to defining one of the attributes (z_i) and relating it to a range of producer costs. The producer costs are directly related to the amount the producer is willing to accept for the particular level of z_i, and, assuming 0 profit (perfect competition) is the same. Thus, this also defines the producer's offer curve for a particular attribute. Using the information provided by the model when a range is given for a variable, the producer would be able to evaluate how profitable (if at all) the project would be, given different outcomes of the variable. For example, assume that the cost of a project is pre-set. A prospective bidder may run the model giving a range for the estimated number of delivered source instructions. This will produce for him a range of estimated costs. From this, the producer can determine if the risk of low profit, or even loss (the cost will exceed the price) is too great given an uncertain input parameter. This scenario can be illustrated using a graph from Rosen's article (fig. 4).

OTHERS

There are many other cost models, each with its own set of assumptions about cost drivers. The previously discussed three are probably among the most widely used and acknowledged in the software cost-estimating field. Other



$\phi 1$, $\phi 2$ are functions for developers 1 and 2, and can be derived from SLIM cost estimation model.

FIGURE 4
DEVELOPERS VIEW OF RISK FROM UNCERTAIN PARAMETERS

models include the Boeing Computer System (BCS) Model (developed by Black and others around 1977), which uses as inputs predicted lines of code for each a five classes. Factors are applied to the estimate for each class giving a total basic effort estimate, which is allocated over 6 program phases. The efforts are then adjusted by application of 9 subjective productivity factors. One criticism of this model is that it is based on outdated (10-15 years old) data. Boehm notes that this model appeared in a report which was trying to discredit it. (The estimates for Boeing projects were off by a factor of five or six.) Another more well-known model is the Doty Model, developed by Herd and others, around 1977. It is of a format similar to the other models in that it utilizes basic equations (two are used, with the equation used depending on the estimated KDSI(thousand delivered source instructions)). They are; $MM = 5.288(KDSI)^{1.047}$, for KDSI>=10, and $MM = 2.060(KDSI)^{1.047}$ (sum f_j, j=1 to 14), for KDSI<10, where f_j are 14 effort multipliers. [8] The problems with this model, pointed out by Boehm, are that there is discontinuity at KDSI = 10, and the f_j factors may be weighted too heavily. [9]

---

8 Parametric Cost Estimating Course Manual (Federal Publications Inc., 1985), p. 166.

9 Barry Boehm, op. cit., p. 514.

V. SOME SIMILARITIES BETWEEN MODELS

Some of the commonalities between the previously mentioned models are apparent. Most models use some measure of estimated lines of code as a major cost driver. Of all the models surveyed in one article, size was the single most important input parameter. [10] COCOMO uses the estimate of the product size KDSI, or 1000 delivered source instructions, in order to determine nominal effort. PRICE S uses the measure of number of machine-level instructions represented by the developed code, input into the variable "INST", as a primary input into the equation. The SLIM model uses number of delivered source instructions as an input to the estimating equation. The estimated size of the program seems to be agreed upon as an important cost driver by most models. The main difference between models seems to be whether program size is estimated by source instructions or object instructions. A chart of factors used by software cost models is included in Boehm's book. It shows that all but one of the models listed use either one or the other parameter. The models included in his survey are: SDC(1965), TRW(1972), PUTNAM SLIM, DOTY, RCA PRICE-S, IBM, COCOMO, SOFCOST, DSN, and JENSEN. The one exception is the GRC(1979) model- it uses number of output formats as the

---

[10] Dr. Robert Setzer, op. cit., p. 603.

sole size parameter. Other size descriptors used, and the number of models (of the 12 surveyed) which used them are: number of personnel(5), documentation (4), number of routines(3), number of data items(3), and number of output formats(2).

Another important factor in most existing software cost estimating models is a measure of the program "difficulty". Again, 11 out of the 12 models surveyed by Boehm used either (or both) the type or complexity of the program as an input parameter. The GRC(1979) model was again the exception, as it used only program language and reuse as program attribute descriptors. Many of the models used more than one variable to describe program attributes. Factors used were; reuse(8), language(6), required reliability(5), and display requirements(3).

Another commonly-used parameter is a computer hardware constraint, quantified by either a time(processing) or storage constraint. One or both of these parameters were used in all but one of the models surveyed. The exception in this case was the SDC(1965) model. Perhaps this is because the SDC (System Development Corporation) model was one of the first estimation efforts, and was not very advanced or accurate in its estimations. The SDC model used the parameters of hardware configuration and development to define the computer attributes. Other parameters used by the models are: development(8), interfacing equipment,

S/W(2), and hardware configuration(2). A fourth category of parameters in most models is a measure of what could be viewed as where the personnel lies on the "learning curve" of the type of programming to be done. This is represented by either the hardware experience or applications experience (or both) in every one of the models surveyed. Other personnel attributes in the models were: language experience(8), personnel capability(6), and personnel continuity(2).

There seemed to be less consensus on the most important project attribute parameters to use. It is hard to distinguish related parameters which covered most of the models, as in the other cases. Either the tools and techniques or the requirements volatility (or both) were used in 11 of the 12 models surveyed, but it is not evident that these two parameters would have a direct relationship. The parameters favored for project attributes were: requirements volatility(9), tools and techniques(8), computer access(8), requirements definition(6), schedule(6), travel/rehosting/multi-site(6), customer interface(4), security(3), and support software maturity(2).

Because these parameters appear, in one form or another, in most of the well-known cost models, it seems they are believed to be among the dependant variables that should be included in an estimation equation. In light of

Rosen's analysis of hedonic price regressions, they are the Zi that should be included in deriving the hedonic prices of software development programs.

Another less obvious common trait of several of the software models is that they have a similar form for estimating the development effort.[11] Seven of the twelve models surveyed by Boehm use the general functional form E = a + bxS^c, where E represents nominal man-months of develpment effort, a, b, and c are constants, and S is a measure of delivered source instructions. The constant c takes a range of values (.91 to 1.2) across the models.[12]

The commonality of this functional form might suggest that, regardless of the additional calibrating factors (for firm-specific development environment, etc.) there is a basic correlation between software "size" and effort, and that they are related in an exponential (versus linear) relationship. Because most of the models are developed using different historical data sources, we can assume that independant research and different data bases have produced this same relationship.

---

11 Dunsmore, Shen, Conte, "A Comparison of a Few Effort Estimation Models," ISPA Conference Proceedings, p. 39.

12 Barry Boehm, op. cit., p. 11.

VI. SOME OUTSTANDING DIFFERENCES

Although the models analysed do exhibit similar characteristics, differences also abound. In Rosen's example, a producer needs to know which attributes (Z_i) are cost drivers, and the weight of each attribute in driving cost (the coefficients of z), in order to produce the optimal (profit-maximizing) bundle of characteristics. In the software cost-estimating models, both of these points are disputed. Which "z_i" should be included depends on which model is used, as is illustrated in Boehm's chart of factors used in models. The functional form of the z_i, and also what their respective coefficients should be, is not consistent between models. Although there do seem to be some areas of general agreement, as explained in the previous section, the hedonic price indices for software development are still not completely defined. A further point of contention between estimators is what should be described by the various "z_i" characteristic levels. In effect, what should the dependent variable be? Some models estimate man-months of effort, others estimate cost of effort, others estimate cost or effort by phase, some define the development effort differently (covering different project phases) than others. This lack of consistency in level of detail and definition across models led to difficulty in comparison in several published cost model

surveys.

VII. USE OF "EX POSTE" FACTORS

Upon examination of several of the cost estimating models, an interesting point arises about the parameters used. In Rosen's article on revealing hedonic price indices, he implies that the specific level of each attribute is known before the market exchange is performed.[13] In software estimation models, some of the parameters used are actually not known until after the transaction (development effort) has occurred. Certainly, some specifications may be made beforehand by the prospective buyer; such as necessary response-time, output formats, and required reliability. Rosen defines the class of goods in his model as being defined by n objectively measured characteristics. Certainly an estimated number of lines of source code, especially in the earlier stages of development, will not be measured objectively, since they cannot be known. There are software sizing models available (e.g., George Bozoki's Software Sizing Model (SSM) but they are not considered to solve the problem of estimating at early stages of development.[14] Thus, the accuracy of the model is hindered by probable misspecification of unknown parameters. The producer may have no say over the specification given to him by the consumer. However, he

---

13 Sherwin Rosen, op. cit., p. 36.

14 Dr. Robert Setzer, op. cit., p. 604.

does have influence over the outcome of the "ex poste" factors. If he realizes that the consumer is calculating a fair price based upon the number of lines of source code delivered, he might certainly be apt to exaggerate that amount in the estimate. This exaggeration could then be justified by actually producing that number, even if a more efficient use of resources was possible. If the original price is based upon an estimated lines of source code, and the producer finds they can accomplish the effort for less, they would not attempt to do so, since they have presumably won the contract. The developer would probably complete the project using the estimated lines of code, and be paid the estimated amount. Since each development program is still fairly innovative, there are not examples to which to compare the effort for accuracy of estimation. An inefficient combination of resources used may be partly a result of using ex poste estimating factors.

By using lines of code as an estimating parameter, one is assuming that it is a positive product attribute. This may certainly not be the case. The fact that the program can handle very complex calculations <u>is</u> a positive aspect. However, using lines of code as a surrogate for the "power" and effort required for the program may be misleading. Lines of code is a function of "power" or effort required only as much as the company is efficient. If the company uses inefficient programming practices (e.g., excessive

documentation, inefficient algorithms), the increase in lines of code represents a "bad" of unnecessary code. It is a "bad" because it costs more to produce, and might very well cost more to maintain (it is very difficult to update programs which exhibit "spaghetti coding" techniques.) Thus, the parameter used as a "good" is actually composed of unknown percentages of lines of code required to accomplish the program efficiently, and a portion of unnecessary code representative of developer inefficiencies. Were it possible to discern the percentages, the lines of code would still be a relevant parameter.

VIII. ACCEPTANCE/RELIABILITY OF MODELS

As noted earlier, intercomparison of the accuracy of various software cost estimating models is difficult, and somewhat subjective, as the definition of the dependant variable and independant variables differ, and also the range of applicability varies between models. Some models have been tailored for certain types of military acquisitions, for example. The Tecolote model was developed to specifically to predict cost and resources needed to develop tactical software in the Navy. It would be biased to compare accuracy of results with another model by testing against private industry cost histories. However, one survey attempted to compensate for differences between models by calculating a "relative RMS" statistic for each which accunts for the average project size in the sample set for scaling.[15] The results are summarized in Table 1.

In their review of several of the popular models, General Research Corp. had some dismal comments about the state of software cost estimating. They noted, "within an environment characterized by similar projects, personnel experience and management techniques, the most accurate models achieved an average estimating error of about 25% on the basis of root mean square error."[16] Also, they

---

15 Robert Thibodeau, op. cit., P. 328.

16 Robert Thibodeau, op. cit., p. 17.

TABLE 1 [17]
RMS ERROR/MEAN PROJECT SIZE

Model Type	Data Set		
	Commercial	DSDC	Sel
Regression			
Aerospace	1.35	2.11	0.605
Doty		1.05	
Farr & Zakorski		16.9	
Tecolote		4.92	
(aIbPT)	0.643	0.933	0.309
Heuristic			
Boeing		0.787	
DoD Micro		1.26	
Price S	0.383	1.44	0.297
Wolverton		0.927	
Phenomenological			
SLIM	0.246	0.216	0.865

[17] $$\text{RMS Error} = \left[1/N \sum_{i} (\text{Act}_i - \text{Est}_i)^2\right]^{1/2}$$

SOURCE: Barry W. Boehm, "Software Engineering Economics," IEEE Transactions on Software Engineering, Vol. SE-10, No. 1, January 1984. p. 10.

concluded, "These result indicate that..there are factors that are not properly accounted for by the models tested."[18] Finally, to fulfill the purpose of the study, they determined that "the best performance obtained by any group of the models tested is not adequate for Air Force needs."[19]

In fact, one is hard-pressed to find even the most successful software models bragging about statistics resulting from empirical tests. More readily available are comments such as, "..this methodology offers significant advantages over the current software estimating procedure."[20] Unfortunately, even if there are some models which are "better than others", all concerned are not aware of which they are. A survey conducted around 1978 found that, within the military ESD environment, that a systematic approach to developing softwre cost estimates within the PO (Program Office) did not exist. Also, some did not perform estimates at all, and others used varying techniques with inconsistent degrees of success."[21] A more recent presentation at the 1986 ISPA Conference noted that "Software cost estimation is complex and still in its

---

18 Robert Thibodeau, op cit., p. 17.

19 Robert Thibodeau, op. cit., p. 18.

20 Dunsmore, Shen, Conte, op. cit., p. 152.

21 Marsha Finfer, et al, Software Acquisition Management Guidebook: Cost Estimation and Measurement (Santa Monica, CA, System Developoment Corporation, 1978), p.25.

infancy as a discipline."[22]

Based upon the review of models in this paper, I feel that one might agree with the previous comments. The lack of consistent data bases, definitions, and approaches to the cost estimating problem for software development leaves the field in a state of confusion and somewhat lacking in credibility. However, the estimating procedures are continually being refined, and perhaps as importantly, the users of the models are becoming more aware of how to use them and how to better estimate the input parameters.

---

22 Dr. Robert Setzer, op. cit., p. 605.

IX. FUTURE RESEARCH

Obviously, there is room for progress in the software cost estimating area. Some suggestions have been made to remove some of the hindrances to developing better cost models. GRC found that the variable which calibrated parameters to the particular development environment (firm) was very important.[23] Unfortunately, most private contractors do not voluntarily give out more data about the internal operation of their company than is essential. Thus, a purchaser (in this case the Air Force) is at a disadvantage in estimating the cost of a project performed by a particular company. Easier access to each organization's particular performance should be allowed to purchasers of software development programs. Personal experience in searching for internal costs for hardware manufacturing programs indicates is that this is not a probable occurrence. In effect, the Air Force needs to have visibility into the offer curves of each of the contending companies for a project. Secondly, it is suggested that data definition and availability be bettered. The Air Force has already taken a step in this direction by establishing the Data and Analysis Center for Software(DACS). GRC recommends that the Air Force take an even more active role in pressing contractors to report development costs, and to

---

23 Robert Thibodeau, op. cit., p. 605.

adhere to a standard set of cost and parameter definitions.

Several studies on cost estimating models arrived at an important conclusion that a cost estimator should use the model most suitable to his/her needs. The model to be used may depend upon application of sotware, amount of detail known, and previous results with the model. Different methods and models of cost estimating may be suited to different cases. Says Boehm, of the varying techniques, "..we should use combinations of the above techniques, compare their results, and iterate on them where they differ."[24] In other words, as the software field is not yet well-defined, the cost estimator should be able to adapt to different situations. This also applies to changing technology. Experts agree that software estimating methods need to account for effects of new technologies, such as the Ada programming language, artificial intelligence, rapid prototyping, and specialized DB languages on future software costing.[25]

Others are already striving to advance the skill of software cost estimating. Dr. Rubin is planning the evolution of "fifth generation expert estimation systems", which will allow for automated intelligence for integrating estimation, planning, and project control. Such knowledge-

---

24 Barry Boehm, op. cit., p. 8.

25 Dr. Robert Setzer, op. cit., p. 600.

based planning aids would allow for continuous estimation and combine it with expert planning and project management control.[26] It appears that this approach seeks to combine the best of two worlds- the trends defined by analysis of historical data collected on projects, and the continual input of an expert in the field. Rubin's idea is to refine the estimate continually, using new data as the project progresses, and using appropriate models for different points. Although this sounds like a new and exciting innovation to the software estimating field, it appears that more emphasis should be placed upon refining the relationships incorporated into the knowledge-based estimating system before software estimating becomes too "high-tech". Then these models could be more confidently integrated into a system where different levels of parameter specification can be used throughout the life cycle. If the above suggestions for attaining better relationships are followed, such a system could eventually be a great boon to corporate and government management of development programs.

---

26 Dr. Howard A. Rubin, 'The Art and Science of Software Estimation: Fifth Generation Estimators," ISPA Conference Proceedings 1985, pp. 5872.

X. CONCLUSIONS

The aim of this study was to examine software cost estimating models as a form of hedonic pricing of software development programs. Examination of the models showed that most models are based, at least in part, on a function relating several characteristics of the software to the estimated effort to develop it. This is equivalent to the $p(z) = p(z_1, z_2, \ldots z_i)$ described in Rosen's article. One difference in the software field, however, is that the z_i are often characteristics which are interrelated by ill-defined ways. The models exhibit an attempt to make that definition. The builders of the data bases use historical data from completed projects to develop their functional relationships. This is equivalent to the observed market prices described in Rosen's article. Using the data bases, the developer is able to estimate the firm's offer function for a given level of profit. They can perform tradeoff studies to determine what effect varying levels of certain attributes will have on costs, and therefore what price will be required to compensate for that cost.

It also is obvious from the above analysis that in a situation such as when the government is contracting for a R&D project under a cost-plus-fixed-fee basis (i.e., the consumer will pay the price of the good without knowing it beforehand), Rosen's analysis is again analogous. In figure 6 the vertical axis represents price, and the horizontal

axis represents level of attribute, z_1. Say, for example that this is the attribute of security in the project. If the consumer were to know the variable β in the offer function for each firm; $\phi = f(z_1, z_2 .. z_i, \pi, \beta)$, where ϕ is the offer amount, z_i are characteristics, π is profit and B is a factor which shifts the curve for each firm. The graph depicts offer curves for varying levels of z_1, given constant profit and optimum values of the other attributes. Perhaps the first firm has lower costs at lower levels of security because some facilities (ex. vaults, security system to prevent unauthorized access to the premises, etc.) are already installed. However, if more stringent security measures were needed by the project, such as secure (windowless) data-entry rooms, and clearances for personnel, the additional costs of constructing the room and processing the clearances would be high. On the other hand, assume the second firm has no facilities, yet the cost of acquiring more strict security measures is less, because a room is easily transformed into a secure data entry room, and personnel are experienced in obtaining clearances. In this case, the government would indeed benefit from knowing the B parameters, because it would be able to see that the more cost-efficient solution would be to choose firm a for a less secure project, and firm b for a more secure one, all other attributes being held constant. The savings from making the

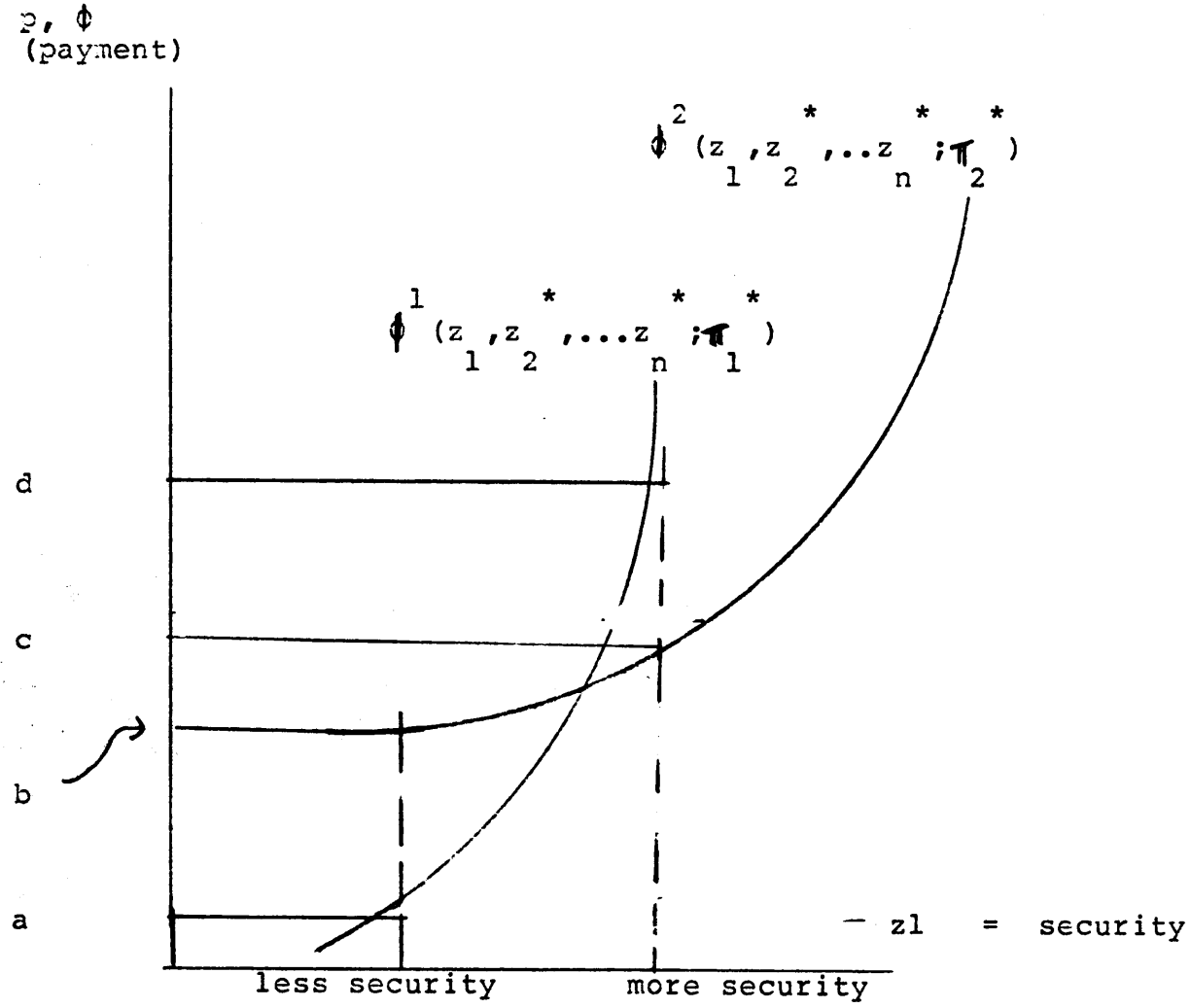


FIGURE 5
HEDONIC PRICES CAUSE PRODUCER SPECIALIZATION

correct choice (assuming a fixed-fee being equal for either firm) would be the amount ab for the less secure project, and cd for the more secure one. This analysis substantiates the recommendation in the GRC study for the Air Force that firm-specific data be more readily accessible to government cost estimators.

As far as the success achieved in defining the hedonic relationship, this study has shown that it is both generally agreed, and apparent from the lack of concurrence among modellers, that there is a long way to go. Cost estimators have yet to prove: relevant independant and dependant variables, proper definitions for the parameters, functional forms, and proper coefficients (weighting) of the variables. One pessimistic assessment of software cost estimating is that there are, "to date, several dozen software cost estimating models... most of which are limited by being proprietary, based on limited data, specific to a single class of projects, inaccurate, not maintained, or lacking automated support."[27] Some points which attained some consistency across models were pointed out. Suggestions for furthering the art/science of estimating were made. Included were: more accurate, widespread recording of development data, more accessability to data, and education in proper use and applicabilities of the models. It was

---

27 Dr. Robert Setzer, op. cit., p. 600.

also suggested that new technologies be constantly evaluated for impact on development costs in general. I feel that this final point is especially important in view of the advances that have been in software only in the last decade. Dr. Rubin, a main proponent of software estimating notes, "... at the same time that research on measurement as a basis for estimation has intensified, the very nature of software development is undergoing revolutionizing changes."[28] If an estimator were to not compensate for this factor, and look only at the project specifications, he/she would be ignoring an important effect on the cost of the project.

It appears that some heroic efforts have been made to create viable software develompent cost data bases, and to develop accurate models from them. However, there do not appear to be published results reflecting tremendous estimating accuracy. Perhaps firms have proprietary models which provide much better results. In any case, it seems that there is opportunity for advancement in the field of deriving a hedonic price model for software development programs that yields statically promising results. I feel that as the proportion of expenditures on software increases, so will the effort put into cost estimation

---

28 Dr. Howard A. Rubin, op. cit., p. 58.

research. Perhaps sooner than one might think, there will be a "fifth-generation" software cost estimating mechanism available to management everywhere. Such developments will hopefully aid in attaining economically efficient solutions in software development programs of the future.

XI. SOURCES CONSULTED

The Analytic Sciences Corporation. The AFSC Cost Estiamting Handbook Series. Reading: The Analytic Sciences Corporation.

Bartik, Timothy J. "The Estimation of Demand Parameters in Hedonic Price Models," Journal of Political Economoy 95 (February 1987): 81-88.

Bilas, Richard A. Microeconomic Theory New York: McGraw-Hill Book Company, 1971.

Boehm, Barry W. "The Constructive Cost Model (COCOMO)." paper presented at the fifth annual meeting of the International Society of Parametric Analysts, St. Louis, MO, 26-28 April 1983.

Boehm, Barry W. "Software Engineering Economics," IEEE Transactions on Software Engineering SE-10 (January 1984): 4-21.

Cheadle, William G. "Software Systems Development Costing and Scheduling Models." paper presented at the seventh annual meeting of the International Society of Parametric Analysts, Orlando, FL, 7-9 May 1985.

Dean, Joseph P. (1Lt., USAF). "Estimating Lines of Code at the Air Force Communication Computer Programming Center." paper presented at the fifth annual meeting of the International Society of Parametric Analysts, St. Louis, MO, 26-28 April 1983.

Dunsmore, H.E., Shen V.Y. and Conte, S.D. "A Comparison of a Few Effort Estimation Models." paper presented at the sixth annual meeting of the International Society of Parametric Analysts, San Francisco, CA, 15-17 May 1984.

Epple, Dennis. "Hedonic Prices and Implicit Markets: Estimating Demand and Supply Functions for Differentiated Products," Journal of Political Economy 95 (February 1987): 59-79.

Ferens, Daniel V. "Software Support Cost Models: Quo Vadis?" paper presented at the sixth annual meeting of the International Society of Parametric Analysts, San Francisco, CA, 15-17 May 1984.

Finfer, Marsha et al., Software Acquisition Management Guidebook: Cost Estimation and Measurement. Santa Monica: System Development Corporation, [1978].

Fox, T. Bernard."An Overview of Software Cost Modeling." paper presented at the 24th International Conference of the National Estimating Society.

Gaffney, John E., Jr., "Estimation of Software Code Size Based on Quantifiable Aspects of Function (with Application of Expert System Technology." paper presented at the sixth annual meeting of the International Society of Parametric Analysts, San Francisco, CA, 15-17 May 1984.

Galorath, Daniel D. "Software Estimating Systems as Management Decision Support Tools." paper presented at the seventh annual meeting of the International Society of Parametric Analysts, Orlando, FL, 7-9 May 1985.

Hamburg, Rodney L. "Evaluation of Software Sizing Models." paper presented at the eighth annual meeting of the International Society of Parametric Analysts, Kansas City, MO, 12-15 May 1986.

Henderson, James M. and Quandt, Richard E. Microeconomic Theory A Mathematical Approach. New York: McGraw-Hill Book Comopany, 1980.

International Society of Parametric Analysts. Proceedings of the International Society of Parametric Analysts Annual Conference, Virginia Beach, VA, 1982.

International Society of Parametric Analysts. Proceedings of the International Society of Parametric Analysts Fifth Annual Conference, St. Louis, MO, 1983.

International Society of Parametric Anlalysts. Proceedings of the International Society of Parametric Analysts Sixth Annual Conference, San Francisco, CA, 1984.

International Society of Parametric Analysts. Proceedings of the International Society of Parametric Analysts Seventh Annual Conferenc, Orlando, FL, 1985.

International Society of Parametric Analysts. Proceedings of the International Society of Parametric Analysts Eighth Annual Conference, Kansas City, MO, 1986.

Jensen, Randall W. "A Comparison of the Jensen and COCOMO Schedule and Cost Estimation Models." paper presented at the sixth annual meeting of the International Society of Parametric Analysts, San Francisco, CA, 15-17 May 1984.

Jensen, Randall W. "Sensitivity Analysis of the Jensen Software Model." paper presented at the fifth annual meeting of the International Society of Parametric Analysts, St. Louis, MO, 26-28 April 1983.

Parametric Cost Estimating Course Manual. Federal Publications Inc., 1985.

Pinsky, Sylvan. "Uncertainty in Software Cost Models." paper presented at the fifth annual meeting of the International Society of Parametric Analysts, St. Louis, MO, 26-28 April 1983.

Rampton, Judy. "Unique Features of the JS-2 System for Software Development Estimation." paper presented at the sixth annual meeting of the International Society of Parametric Analysts, San Francisco, CA, 15-17 May 1984.

RCA PRICE Systems. "A Presentation of the RCA-PRICE Cost Models to the Air FOrce Managers at Wright Patterson Air Force Base Dayton, Ohio Wednesday, June 25, 1986."

Rosen, Sherwin. "Hedonic Prices and Implicit Markets: Product Differentiation in Pure Competition," Journal of Political Economy 95 (January/February 1974): 34-55.

Rubin, Dr. Howard. "The Art and Science of Software Estimation: "Fifth Generation Estimators." paper presented at the seventh annual meeting of the International Society of Parametric Analysts, Orlando, FL, 7-9 May 1985.

Setzer, Dr. Robert. "Spacecraft Cost Estimation: Striving for Excellence Through Parametric Models (A Review). paper presented at the eighth annual meeting of the International Society of Parametric Analysts, Kansas City, MO, 1986.

Singhal, Madhu. "Software Size Estimator." paper presented at the eighth annual meeting of the International Society of Parametric Analysts, Kansas City, MO, 12-15 May 1986.

Thibodeau, Robert. An Evaluation of Software Cost Estimating Models. New York: Rome Air Development Center, [1981].

The vita has been removed from the scanned document