Winter 1998

An Alternative Method for Determining Adjusted Function Points as the Basis for Software Cost Estimating

William Alexander Eldred
Old Dominion University

AN ALTERNATIVE METHOD FOR DETERMINING ADJUSTED FUNCTION POINTS AS THE BASIS FOR SOFTWARE COST ESTIMATING

by

William Alexander Eldred
B.S. June 1962, United States Naval Academy
M.S. June 1972, Massachusetts Institute of Technology
Ocean E. June 1972, Massachusetts Institute of Technology

A Dissertation Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirement for the Degree of

DOCTOR OF PHILOSOPHY

ENGINEERING MANAGEMENT

OLD DOMINION UNIVERSITY
December 1998

Approved by:

Resit Unal (Director)

Derya A. Jacobs (Member)

Abel A. Fernandez (Member)

C. Michael Overstreet (Member)

ABSTRACT

AN ALTERNATIVE METHOD FOR DETERMINING
ADJUSTED FUNCTION POINTS AS THE BASIS
FOR SOFTWARE COST ESTIMATING

William Alexander Eldred
Old Dominion University, 1998
Director: Dr. Resit Unal

As software costs become an increasingly higher percentage of total computer system costs, it becomes increasingly more important for software development managers to have the ability to predict development costs with reasonable accuracy early in the software development cycle. Software development cost estimates are based in large measure on software size. Function points are considered by many to be a de facto industry standard as a size metric. The function points technique, unlike lines-of-code, can be applied early in the software development cycle and is language independent. Critics claim that the function point "value adjustment factor," which purports to capture the effects of software complexity considerations in the final function point count, is inadequate as currently determined. The research described herein develops a new approach, using a less restrictive multiplicative model instead of the existing additive model, for calculating the value adjustment factor. The proposed approach was implemented and evaluated using data from 301 software development projects. However, no improvement in model performance was realized, at least using the limited data available, which were for the most part representative of only one software application domain. Therefore, no conclusion can be drawn from the results of this effort as to whether the proposed multiplicative model is an improvement over the additive model for software development projects in general. The results do, however, focus attention upon two areas which merit further investigation: the performance which would be realized in applying the proposed new model to data more representative of a cross-section of modern software; and the question of whether the function point general system characteristics (upon which the value adjustment factor is based), as currently defined, adequately capture the effects of potential system cost drivers. This research makes the following contributions: an alternative approach for capturing the effects of Function Point general system characteristics on development effort and cost has been demonstrated; there is an indication that the general system characteristics are in need of a thorough review for appropriateness and adequacy; and the need for better dialogue between various elements of the software sizing and cost estimating community is identified.

To my wife Judy, without whose love, encouragement, and patience the successful completion of this undertaking would not have been possible.

ACKNOWLEDGMENTS

I wish to express my sincere thanks to those individuals who have contributed in some way to the successful completion of this effort. Foremost among these is Dr. Resit Unal, whose interest, encouragement, timely suggestions, and sense of humor provided the support I needed to see my work through to completion. I also wish to thank the other members of my committee, Dr. Derya Jacobs, Dr. Abel Fernandez, and Dr. Mike Overstreet, for their time, interest, feedback, and suggestions.

I would also like to thank those practitioners and researchers in the software metrics and cost estimating community who contributed, in ways large and small, to my ability to define the research issue and obtain the necessary information to proceed. These include: Bryan Piggott of PRC Inc.; Ray Madachy of Litton Data Systems and also part of Barry Boehm's team at the University of Southern California; Rob Donnellan; Denis St-Pierre and Jean-Marc Desharnais of the Software Engineering Management Research Laboratory at the University of Quebec at Montreal; Paula Jamieson, Andrew Sanchez, and Gordon Lundquist of the International Function Point Users Group (IFPUG), as well as the IFPUG staff in Westerville, Ohio; Chris Kemerer of the

University of Pittsburgh; Dick Stutzke of Science Applications International Corporation; Capers Jones of Software Productivity Research, Inc.; and Dillard Boland of Computer Sciences Corporation. I would like to thank Jim Duff for his interest and for his advice regarding the statistical aspects of my work. Thanks are due also to the PRC Inc. Technical Library staff in McLean, Virginia - Barbara Kopp, Alice Hill-Murray, and Pat Garman - for their persistence in helping me locate needed materials. I would also like to thank my colleague and co-worker, Dr. J. Terry Ray of PRC Inc., for sharing the insight he gained as he pursued a similar goal.

I owe a special debt of gratitude to the late Richard A. Bihr, whose encouragement and own unending desire to learn helped rekindle my interest in and enthusiasm for the attainment of academic objectives.

Finally, I would like to express my thanks to my daughter, Alina Dawson Eldred, whose own success in the pursuit of academic excellence set a superb example and helped her Dad not lose sight of his goal.

TABLE OF CONTENTS

LIST OF TABLES

CHAPTER I

INTRODUCTION

Background

As late as the early 1970s, software costs accounted for less than 40 percent of total computer system costs, and hardware was the dominant cost factor. Since that time, hardware costs have steadily decreased, while the costs of software development and maintenance have increased significantly. The result is that software costs now account for about 90 percent of the total system cost to the end user over the life cycle of the software (Wellman 1992, 30). This has focused substantial attention on efforts to predict and control software development costs.

Reasonably accurate early estimation of software development costs is important to software managers for several reasons. These include: the need for orderly allocation of personnel and other resources; the making of "go/no-go" decisions, based on cost, of whether or not to proceed with a software development effort; and, in bidding on a contract or quoting a price to an outside customer, the need to avoid substantial cost overruns or underruns. Overestimating can result in failure to win

---

Style specifications follow the Old Dominion University Dissertation Guide and A Manual for Writers of Term Papers, Theses, and Dissertations, by Kate L. Turabian, where applicable.

the contract in the first place, while underestimating, depending on the type of contract, can result in a "bottom line" loss to the software development organization or a very unhappy customer when asked to pay a substantially higher price than first quoted. Software users (customers) have shown an increasing unwillingness to accept schedule slippage and cost overruns unless the developers bear an increasing share of the resultant penalties (Srinivasan and Martin 1994, 70).

Clearly, then, the financial success of a software development project depends, in large measure, on the ability of the software manager to estimate the cost of software development reasonably accurately, prior to the start of the project (Navlakha 1990, 255). As late as 1996, Garmus and Herron state that "unfortunately, a method of doing so [providing timely and accurate project estimates] early in a project has not been adequately addressed and standardized within the software industry" (Garmus and Herron 1996b, 57).

Early software cost estimating efforts were based principally on expert judgment or, where it was available, historical information for software development projects similar, or analogous, to the development project being estimated. Such historical-experiential models were not particularly reliable, as their accuracy depended on the experience of the experts with similar projects or the currency and accuracy of the historical data being used as the basis for the estimate and the degree to which the project being estimated had features in common with those from which the historical data had been gathered.

In the 1970s a number of algorithmic software cost estimating models began to emerge. These models, which generally used mathematical algorithms, were regarded by many as true models, whereas the earlier (historical-experiential) approaches were considered more as methods (Wellman 1992, 36). Conte, Dunsmore, and Shen (1986, 279-330) subdivided these into statistically based methods, theoretically based models, and composite models, providing a discussion and examples of each. They described composite models as incorporating "a combination of analytic equations, statistical data fitting, and expert judgment" (300).

Probably the best known of the composite algorithmic models was Barry Boehm's Constructive Cost Model, or COCOMO, which Boehm introduced in 1981 (Boehm 1981). Boehm presented three levels of COCOMO: basic, intermediate, and detailed. Basic COCOMO was represented by a formulation typical of the algorithmic models (Wellman 1992, 36), specifically:

$$\mathrm{MM} = aL^{b} \tag{1.1}$$

where MM is manmonths of effort, and *L* is lines of code. Intermediate COCOMO introduced the effect of 15 cost drivers (in addition to the primary cost driver, size, defined by lines of code). Detailed COCOMO provided for a breakdown of the estimate by phase of the software development cycle.

As did most of the algorithmic models developed through the mid-1980s, COCOMO used as the measure of a software project's size the number of delivered source instructions (lines of code), or DSI, expressed as KDSI (K = thousand). (Other models referred to this measure as source lines of code (SLOC) or KSLOC.)

The use of lines of code as a measure of software size presented two significant problems, especially for use of the models early in the software development cycle. At that point in the cycle, a reasonably accurate prediction of project cost would be of value to a software development manager in quoting cost and schedule estimates to a customer or in planning the allocation of resources. The first of these problems is illustrated by Jones in his discussion of what he calls the "paradox of reversed productivity for high level languages" (Jones 1991, 53). This phenomenon manifests itself as follows: as the computing power of programming languages improves (i.e., on the average, the amount of computational instruction contained a line of code increases), real economic software productivity should improve, even though there will probably be some increase in the effort to produce one line of code in a more sophisticated language. However, any software metric which is based on lines of code will show an increase in the effort (cost) per line of code, making it appear that productivity is worse, not better. Since a single line of code, on the average, could have a different computational value depending on the level of the language, source lines of code does not represent the same (or even substantially the same) value for all software development efforts. Thus, SLOC broke down as a usable size metric.

The second problem with using lines of code as a size metric was that, by the time the number of SLOC was known with a reasonable degree of accuracy, a commitment to a project had already been made and development was well underway. Indeed, Mukhopadhyay and Kekre (1992, 915) maintain that "LOC is not known with reasonable degree of certainty until after programming is completed." While this was useful for assessing productivity, i.e., determining what a project should have cost relative to what it actually cost, models based on lines of code had little or no value in a prediction role for early cost estimating to support go/no-go decisions, price quotes to a prospective customer, resource allocation planning, and the like.

At about the time that Boehm was developing COCOMO, Allan Albrecht (1979) of IBM was introducing a measurement technique which he called "function points" that he and some colleagues had developed at IBM. According to Gaffney (1996, 2), who, following the introduction of function points, collaborated with Albrecht on at least one paper concerning the validity of function points, the function points approach was created "to improve the communication between IBM developers of custom commercial software and their customers concerning the requirements for prospective software systems." Gaffney goes on to say that "A key problem in software development is how to state requirements in such a manner that both software system providers and acquirers can understand them. Albrecht devised the function point measure to enable his customers to state their requirements so that they could be readily translated into a cost estimate by the IBM development team" (1996, 2-3). Albrecht himself stated as an objective of the work which resulted in the development of the function point approach "to develop a relative measure of function value delivered to the user that was independent of the particular technology or approach used" (Albrecht 1979, 84). In the presentation in which he introduced function points, Albrecht appeared to be focusing on the measurement of productivity as much as software project cost estimation. Indeed, it was titled "Measuring Application Development Productivity" (Albrecht 1979). Nonetheless, it had resulted in introducing a software size metric that was independent of the programming language and which could be determined at the point in the software development cycle at which the system requirements were fully defined, or specified.

In the research that culminated in the establishment of the function points approach, Albrecht and his associates found that the basic value of a software application's function "was consistently proportional to a weighted count of the number of external user inputs, outputs, inquiries and master files." Albrecht weighted the counts thus obtained by numbers "designed to reflect the function value to the customer. The weights used were determined by debate and trial." Advising that the weights as shown below had given them "good results" (1979, 85), Albrecht presented the following as the initial set of calculations:

Number of Inputs × 4

Number of Outputs × 5

Number of Inquiries × 4

Number of Master Files × 10 (1979, 85)

The total so obtained was then adjusted for the effect of ten "complexity" factors. This complexity adjustment could result in an increase or decrease in the unadjusted total of as much as 25 percent. The final result was (in Albrecht's words) "a dimensionless number defined in function points which we have found to be an effective relative measure of function value delivered to our customer" (1979, 85).

In 1984 Albrecht issued a major revision to the function points counting algorithm presented above (Jones 1991, 60-64) which resulted in its being more complicated (and presumably more accurate). A fifth calculation was added (Number of External Files Referenced), and the number of weighting factors was expanded to allow for three different weights which can be applied to each of the five types of count. Which of the three different weights to use depends on the complexity (low, average, high) of the individual entity being addressed. The ten original system complexity factors were expanded to fourteen. These are now referred to as "general system characteristics" (Garmus and Herron 1996a, 81-90). The values of the 14 general system characteristics are used to determine a "value adjustment factor," which, when applied to the unadjusted function point total, can result in an increase or decrease of as much as 35 percent.

Following its introduction, the function points approach began to increase in use and popularity. In 1983, the association of IBM's commercial clients, GUIDE, established a working group on function points (Jones 1991, 81). By 1986, several hundred companies, many but not all of them clients of IBM, were using the function points approach. In 1986, the International Function Point Users Group (IFPUG) was formed as a nonprofit organization devoted exclusively to the utilization of function points and the dissemination of data derived from function point studies. IFPUG has evolved into a major association concerned with all aspects of software measurement. According to Jones (1991, 81), "the IFPUG counting practices committee has become a de facto standards group for normalizing the way function points are counted, and it has done much to resolve the variations in terminology and even the misconceptions that naturally occur when a metric gains wide international use." IFPUG publishes a Function Point Counting Practices Manual to provide detailed counting rules, ensure that the counting rules are updated to encompass new programming techniques and languages, and thereby to ensure standardized application of the function points approach. The current version of the Manual is the fourth (Version 4.0). IFPUG also oversees the training and certification of analysts in the counting of function points.

Kemerer (1987) and others performed validation studies of the function points approach and generally found correlation with actual effort better than with other software estimating techniques. The function points approach, however, has not been without its critics. Many (Symons 1988, 10; Keyes 1992, 44; Matson, Barrett, and Mellichamp 1994, 276; Srinivasan and Martin 1994, 73; Whitmire 1995, 43) felt, since the function points approach was developed for, and using data from, management information (or "business") systems, that it was inadequate for use as a size metric for software development in areas such as scientific or technical systems, communications software, "real-time" systems, computer operating systems, graphical user interfaces (GUIs), and object oriented programming. The primary criticism in the literature was that the function points approach did not adequately capture the complexity of software development in these areas. This concern led to the development of variants of the function points approach. These were introduced under such names as feature points, function points Mark II, 3D function points, and object points. The basic function points approach continues, however, to enjoy widespread use and the advantages of a formalized user group.

Additionally, it is important to realize that the function points approach has been evolving since its introduction. As mentioned earlier, even before the establishment of IFPUG in 1986, Albrecht had made some improvements in his approach (Jones 1991, 60-64). Since IFPUG had released its fourth revision to the <u>Function Point Counting Practices</u>

Manual within ten years of the organization's founding, it is likely that the changes to the counting practices (or "rules") reflected in succeeding revisions were made, at least in part, in response to the criticism. Many argue now that the function points approach indeed does have applicability and produces valid results in the non-business areas cited earlier (Bernstein and Lubashevsky 1995, 17; Garmus and Herron 1996a, Chapter 11). In reviewing the literature it appears that, though some shortcomings to the method may remain, the function points approach currently represents the available methodology closest to an industry standard for software size estimation. As noted earlier, it is language-independent, and a reasonably accurate function point count can be determined "early on," prior to beginning actual software development.

As the function points approach was evolving to its current status within the industry, research was continuing which built upon the original COCOMO model. In 1995, Boehm and researchers at the University of Southern California introduced an updated version of COCOMO, designated COCOMO II (Boehm et al. 1995). COCOMO II, like the original version, is intended to be publicly available. There is an Early Design variant of the model, specifically intended for use as a development effort predictor. Although it remains principally a lines-of-code based model, COCOMO II does permit the acceptance of function points as a software size metric input.

Purpose

The purpose of this research effort is to develop and evaluate an alternative to the current approach to determining the function point "value adjustment factor." This factor is applied to the unadjusted function point count for a software development project in order to arrive at the adjusted, or final, function point count. The value adjustment factor aggregates the values of the 14 general system characteristics, and it is then used to "adjust" the unadjusted function point count to produce the adjusted count. It therefore can significantly affect the value of the adjusted function point count. As calculated using the current approach, the value adjustment factor can result in an increase or decrease of as much as 35 percent in converting the unadjusted function point count to the adjusted function point count. Consequently, research which seeks a better way of determining the value of the function point value adjustment factor (in terms of producing an adjusted function point count which is more closely correlated with development effort and therefore cost) is highly desirable if function points are to be utilized as the basis for predicting or estimating development cost.

The current approach has received substantial criticism, with some maintaining that little value is added by making the adjustment to the raw, or unadjusted, function point value (Kemerer 1987, 428; Gaffney 1996, 7; Boehm 1997, 25-26; Finnie, Wittig, and Desharnais 1997, 43-44; Jamieson 1997). A review of the literature indicates that the treatment of software complexity considerations and other "effort multipliers" by the COCOMO II model is of a form different from that used to reflect similar considerations in the value adjustment factor, and consequently the adjusted function point count, based on the values of the 14 general system characteristics. As a result, a different approach appears to be needed for determining the function point value adjustment factor. This research attempts to eliminate constraints of the current method (which are the basis for some of the current criticism) and develop an alternative approach to produce adjusted function point values which will be more closely correlated with development effort and cost than is currently the case. The objective is to improve the use of function points as the basis for predicting or estimating development costs and to contribute to the literature on the use of function points in software cost estimating.

CHAPTER II

LITERATURE REVIEW

Survey of Literature on the Problem

The volume of literature which exists on software metrics, and in particular on that aspect of software metrics dealing with size, effort and cost estimation, is quite large and growing. The discussion herein will deal with the subset of that volume of literature, relevant to the problem stated above, which addresses: the introduction of the original COCOMO as a cost estimating approach and of function points as a software size metric alternative to lines of code; the debate and ultimate acceptance by a substantial portion of the community of the function points approach as a de facto industry standard for measuring software size; the discussion of the degree of internal software complexity reflected in a function point count plus how complexity is taken into account in estimating effort from function points; the general nature of estimation models which base their estimates on function points; and, finally, the introduction of COCOMO Version 2.0 (COCOMO II).

The Introduction of the Function Points Approach

Allan Albrecht introduced the function points approach in a paper he presented at a symposium in October, 1979, in Monterey, California. The main focus of Albrecht's paper was on software development productivity and how to measure and then compare productivity among development projects. In the paper he states:

> To measure productivity we had to define and measure a product and a cost. The product that was analyzed was function value delivered. The number of inputs, inquiries, outputs, and master files delivered were counted, weighted, summed, and adjusted for complexity for each project. The objective was to develop a relative measure of function value delivered to the user that was independent of the particular technology or approach used. (Albrecht 1979, 84)

He then presents the weighting values for each of the four counts, as discussed earlier herein, advising that the weights indicated had given them "good results." He follows that with a discussion of the manner in which above or below average complexity was addressed: "If the inputs, outputs, or files are extra complicated, we add 5%. Complex internal processing can add another 5%. On-line functions and performance are addressed in other questions" (85). The final outcome was that the total count could vary by as much as plus or minus 25 percent from its unadjusted value when such adjustments for complexity were made. The resulting "dimensionless number defined in function points" was found by Albrecht to be "an effective relative measure of function value delivered to our customer" (85).

For a measure of cost, Albrecht used work-hours (alternatively described in the literature as manhours or person-hours). He cautions that it was important to include the whole software development process, including the design phase, in the measurement in order to draw meaningful conclusions.

The Introduction of COCOMO

In 1981, Barry Boehm published his book titled Software Engineering Economics, which can appropriately be described as a "landmark" work in the field of software cost estimating. In it he discusses his motivation for publishing the book, provides a detailed discussion of the economic aspects of software engineering, discusses the underlying theory behind development of the model, and, of course, introduces the three levels of the COCOMO model as described earlier. He then discusses application of the COCOMO model in managing software development.

Since its introduction, COCOMO has become probably the best known of the lines-of-code based cost estimating metrics. However, Boehm is quite frank about its limitations. He devotes a whole chapter to "Factors Not Included in COCOMO." Factors excluded from the original COCOMO Model include the type of application, the level of the programming language, complexity, and others. In each case, Boehm explains the rationale for not including a given factor. It is, however, likely that the significance of at least some of the excluded factors is greater now than it was in 1981.

Validation of Function Points as a Software Size Metric

In a paper published in November of 1983, Albrecht and Gaffney describe their work which had as an objective the validation of the function points approach by examining the correlation between function points and SLOC as well as between function points and development effort. In their words:

> The thesis of this work is that the amount of function to be provided by the application (program) can be estimated from an itemization of the major components of data to be used or provided by it [this alludes to the function point calculations discussed earlier]. Furthermore, this estimate of function should be correlated to both the amount of "SLOC" to be developed and the development effort needed. (Albrecht and Gaffney 983, 639)

They point out the advantages of the function points approach in that it can provide a measure of software size relatively early in the development cycle (based on information available from dialogue with the user and from the statement of basic requirements for the software) and that the resulting function point count relates to user requirements in a way that is more easily understood by the user than is SLOC.

Based on their work, Albrecht and Gaffney conclude that "at least for the applications analyzed, both the development work-hours and application size in 'SLOC' are strong functions of 'function points'... (644)." They observe that "it appears that basing applications development effort on the amount of function to be provided by an application rather than an estimate of 'SLOC' may be superior" (644). However, they suggest that, until a sufficient supporting base of productivity data could be developed to support such direct estimating, a "two-step" process could be adopted which would use function points to estimate, early in the development cycle, the SLOC to be produced. The work effort would then be estimated from the estimated SLOC.

In 1987, Kemerer conducted an empirical validation of four algorithmic models used in software cost estimation. The validation was accomplished using simple linear regression analysis and was based on both correlation of the models' output with actual effort expended and the magnitude of the relative error (MRE) between the model outputs and actual effort expended. Included in these were COCOMO and function points. For function points, Kemerer used models previously developed by Albrecht to predict man-months from function points as well as to predict SLOC from function points and man-months from SLOC. In his conclusions, Kemerer states that "Albrecht's model for estimating man-months of effort from Function Points has been validated on an independent data set." He advises that the results of his analysis "seem to validate Albrecht's claims that Function Points do correlate well with eventual SLOC" (1987, 424-425).

In 1990, Low and Jeffery published the results of an empirical research project into the consistency and limitations of function points as an a priori measure of system size compared to the traditional lines of code measure (Low and Jeffery 1990). Based on their analysis, they concluded that "function point counts appear to be a more consistent a priori measure of software size than source lines of code" (1990, 71). They therefore recommended that function point estimates be used in preference to lines of code estimates as the measure of system size, for the type of software investigated in their analysis (business applications), when estimating a priori the effort required for application development. In a subsequent study, Kemerer and Porter identified the source and impact of such inter-rater variations in the application of function point counting rules, suggesting that the results of their analysis could "provide guidance to function point standard setting bodies [e.g., IFPUG] in their deliberations upon rule clarification, and to practitioners as to where the difficulties lie in the [then] current implementation of function points" (1992, 1021). In the 1993 report of the results of a field experiment on "reliability of function points measurement," Kemerer concluded that "this experiment has shown, contrary to some speculation and the limited prior research, that ... the interrater ... reliability of function points measurement [is] sufficiently high that their reliability should not pose a practical barrier to their continued adoption and future development" (Kemerer 1993, 96).

Generalizability of Function Points

One of the early criticisms of the function points approach was that it was only suitable for use with management information or "business" systems. Critics have included Symons (1988, 10), Rubin (Keyes 1992, 44), Matson, Barrett, and Mellichamp (1994, 276), Srinivasan and Martin (1994, 73), Whitmire (1995, 43), Glass (Oskarsson and Glass 1996, 112), Major (1996, 4-9), and, to some extent, Jones (1991, 81-82). The debate continues as to the appropriateness of the function points approach in other "application domains" (e.g., scientific and technical software, "real-time" systems, system software, communications software) or modern programming techniques (e.g., graphical user interfaces and object-oriented programming). To address these perceived shortcomings of the function points approach, some "off-shoots" of function points have been proposed, among them function points Mark II, 3D function points, feature points, and object points. Others, however, especially recently, point to the success of the function points approach in a wide range of applications (Bernstein and Lubashevsky 1995, 17; Garmus and Herron 1996a, Chapter 11). Presumably IFPUG has been modifying and expanding its published counting practices to accommodate a wider range of application environments. Jones (1996b, 7) recommends eight practical criteria for those considering the selection of metrics for measuring software productivity and quality and observes that "it is interesting that the Function Point metric is currently the only metric that meets all eight criteria." Further, recent dialogue suggests that function point analysis is being used as the basis for a set of software development standards being developed by the International Organization for Standardization (ISO) (Rehesaar 1997, 1).

Although the software metrics literature is not unanimous on the generalizability of function points, it appears clear that the function points approach comes closer than any other metric to being an industry-wide standard for the measurement of software size.

Software Complexity Considerations in Function Point Analysis

The function point counting process considers two levels of software complexity: the first is that which determines the weighting factor to be applied to each of the five entities which are counted as the first step in the function point counting process. Accepted function point counting practices provide the rules for determining whether the "low," "average," or "high" weighting factor should be used. Albrecht's original model (Albrecht 1979, 85) did not make the distinction between low, average and high complexity of individual entities. The second category is that of the complexity intended to be captured by the 14 "general system characteristics" (Albrecht originally had ten) which results in an adjustment factor applied to the raw, or unadjusted, function point count to produce the final, or adjusted, function point count. To the extent that complexity effects are not captured at these two levels by applying prescribed function point counting procedures, they must somehow be otherwise incorporated in converting the function point size estimate into a reasonably accurate effort or cost estimate.

The adequacy of the general system characteristics and the resultant value adjustment factor, as it is currently determined, in capturing the effects of complexity has been questioned. Kemerer (1987, 424), for the data used in his validation study, observes that "the difference between using Function Points, which include 14 factors that modify the Function Counts, and the Function Counts themselves, seems slight in this instance." Symons (1988, 4) states that "the restriction to 14 factors seems unlikely to be satisfactory for all time" and "the weights ('degree of influence') of each of the 14 factors are restricted to the 0-5 range, which is simple, but unlikely to be always valid." Gaffney (1996, 8) reports on research which indicates that "an estimate of effort based on counts of only one or several of the function point primitives [referring to the five entities which are the initial basis for counting function points] can be as accurate as an estimate based on function points. Thus, these results suggest that only some of the elements of a function point count need be obtained in order to develop good estimates of development effort." Boehm (1997, 25), in referring to the general system characteristics, states that "each of these fourteen characteristics ... thus have a maximum of 5% contribution to estimated effort. This is inconsistent with COCOMO experience." Finnie, Wittig, and Desharnais (1997, 44) conclude that "the VAF [value adjustment factor] appears to be inadequate and different methods of adjusting an estimate to account for complexity need to be devised." Jamieson (1997), in suggesting areas in which research might be useful, indicated to this researcher that many in IFPUG question the value of the general system characteristics and the value adjustment factor.

Regarding complexity considerations beyond obtaining the final function point count (i.e., not captured by the value adjustment factor), Garmus and Herron (1996b, 58), after stating that "to some extent, complexity levels are evaluated by the function point 14 general system characteristics," advise that "The assessment of a project's complexity must also take complex interfaces, database structures, and contained algorithms into consideration. You can assess this by using the following five levels of complexity" [they then list characteristics representative of the five levels as they have defined them]. Jones (1996a, Chapter 4) similarly alludes to additional software complexity which should be taken into account in assessing factors to be considered in translating the final function point count into an estimate of software development effort. He identifies three forms of complexity: (1) the complexity of the underlying problem and algorithms; (2) the complexity of the source code; and (3) the complexity of the data and data structures. He advises "when the major forms of complexity that affect software projects are considered, there are at least 20 of them," going on to list and discuss each of the 20. Dreger (1989, Afterword) includes "tools" and "techniques" as project attributes which must be considered in estimating project work effort from function points. What appears clear here is that current counting practices do not, in most cases, result in a final function point count which adequately captures the effects of all software complexity factors.

Existing Techniques and Models for Deriving Effort Estimates from Function Point Counts

Current techniques and algorithms which use function points as the basis for software size measurement and cost and schedule estimation (i.e., which incorporate cost and schedule driver factors not captured in the adjusted function point count) are for the most part proprietary. Giles and Barney (1995, 8-10) list ten automated metrics tools used in software cost estimation. Five of these accept function points as an input. Four of these five are proprietary and therefore require the purchase of user licenses which range in cost (as of 1995) from 15,000 to 20,000 dollars. The remaining estimation tool is government owned and therefore available at no cost for government organizations, but no mention is made of its being available to private companies. To this researcher's knowledge, outside of the very recently introduced COCOMO II, there exists no readily available, non-proprietary method or tool for producing total software development effort (manhour) estimates based on function points.

COCOMO II

In a paper published in the Annals of Software Engineering (Boehm et al. 1995), Barry Boehm and his associates announced the forthcoming release of COCOMO Version 2.0 (since redesignated COCOMO II).

Boehm advises that "the major new modeling capabilities of COCOMO 2.0 are a tailorable family of software sizing models, involving Object Points, Function Points, and Source Lines of Code; nonlinear models for software reuse and reengineering; an exponent driver approach for modeling relative software diseconomies of scale; and several additions, deletions, and updates to previous COCOMO effort-multiplier cost drivers" (1).

In the paper, Boehm describes his "future software practices marketplace model," in which he presents his vision of the software marketplace into the twenty-first century and divides the marketplace into five sectors, reflective of the various forms which state-of-the-art software development can take. The COCOMO II model is designed for applicability to this vision of the software marketplace. In other words, COCOMO II is designed for use with current and projected software development practices.

It is Boehm's stated intention that the COCOMO II model will be publicly available. In Chapter 2 of the COCOMO II Model Definition Manual, he says (1997, 5): "COCOMO II follows the openness principles used in the original COCOMO. Thus, all of its relationships and algorithms will be publicly available." As was indicated above, the model has sufficient flexibility to permit (although not require) the acceptance of function points as an input. Additionally, Boehm has defined an "Early Design Model" as a variation of the full COCOMO II (Post-Architecture) Model. Nonetheless, claims of availability and flexibility aside, COCOMO II remains substantially a lines-of-code based model, as was the original COCOMO, with "rule of thumb" conversion factors used to convert function points to nominal lines-of-code values prior to using the model.

Of interest for purposes of the research proposed herein is the manner in which the COCOMO II algorithm treats cost driver factors in order to capture their impact on development effort and cost. It is fundamentally different from the manner in which the current function points approach captures the influence of the 14 general system characteristics in determining the adjusted function point count. The COCOMO II approach suggests a possible improvement to the current function points approach.

Summary of the Literature Review

In this chapter, the literature which pertains to the introduction of function points as a proposed software size metric and of the original COCOMO as a lines-of-code based cost estimating model was discussed. Following this was a discussion of the literature which validates function points as an industry-wide size metric, at least for certain application domains. Next, the debate over the generalizability of function points as an "industry standard" for software sizing was discussed. Following this was a discussion of the treatment of software complexity considerations used in the function points approach, including criticism which questions the adequacy of the current method, in terms of its ability to capture the impact of software complexity factors, for calculating the function point value adjustment factor and consequently the final (adjusted) function point count for a software development project. This was followed by a brief discussion of the existing methods and algorithms, most of them proprietary, used to estimate development cost from function points. Finally, the recently developed COCOMO II was introduced. Of particular interest to this research effort is the fact that COCOMO II postulates an approach for calculating the impact of software development cost drivers, including complexity factors, which is fundamentally different from the manner in which the influence of the 14 general system characteristics is captured in determining the final function point count. This suggests that there may, based on the treatment of cost drivers in COCOMO II, be an alternative approach for capturing the effect of the 14 general system characteristics in determining the adjusted function point count which will answer at least some of the criticism levied at the current approach.

CHAPTER III

RESEARCH ISSUES

The Problem

As indicated in the literature review, function point analysis is probably the most widely used technique for measuring software size, and function points can arguably be considered an industry standard as a size metric. However, function points do not do a good job of reflecting, or "capturing," all software complexity factors and the effects of other system characteristics.

The function points approach first attempts to capture a measure of complexity by assigning a complexity level (simple, average, complex) to the five basic entities (inputs, outputs, inquiries, internal files, and external files referenced) which are counted as a first step in determining a software project's function point count. These are weighted and their weighted values summed to produce the raw, or unadjusted, function point count.

Next, an attempt is made to capture the effects of 14 "general system characteristics," whose values are assessed and used as the basis for a "value adjustment factor" which is applied to the unadjusted count, resulting in a final, or adjusted, function point count. However, much additional manipulation is needed in order to use the adjusted function point count as a practical basis for software development cost estimation, resource allocation, and workload forecasting. This manipulation is embodied in the various cost estimating techniques which estimate development cost using function points as a basis (Giles and Barney 1995, 10).

The literature reflects the feeling of many that the function point "value adjustment factor," or VAF, which is calculated from the values of the 14 general system characteristics (GSCs) and applied to the raw or unadjusted function point count to produce the final or adjusted function point count, is inadequate in that it adds little value toward cost or effort estimating, at least in its current form. It is generally accepted, however, that there needs to be a method for capturing the impact of complexity and other factors, beyond simply the size metric (function points or lines-of-code), in estimating development effort and cost.

Throughout the discussion contained herein, there are two underlying assumptions regarding use of the terms "effort" and "cost": one is that by far the greatest contributor to software development costs is the labor involved. The other is that if one can estimate development labor in manhours (or manmonths, manyears, etc.) accurately, then, by knowing the labor rates, labor distribution, and cost accounting structure for one's organization, one can readily convert manhours to dollar costs for that organization. In this sense, the terms "effort" and "cost" are treated as being interchangeable.

However, a consistently applied definition of what does and does not constitute development effort, upon which to base development cost, is needed, rather than leaving this distinction up to the discretion of the individual practitioner. Boehm (1981, 51-52), in discussing the software life cycle work breakdown structure, provides relatively detailed guidance in this regard. He does not include the requirements development function as part of the software development process, but he does include such functions as design, coding, testing (at all levels through acceptance testing), documentation, configuration management, quality assurance, and documentation development, as well as the management function. The exact boundaries of the definition of what constitutes development effort are not as important as is general agreement on what those boundaries are.

Boehm et al., who developed the COCOMO II model, concur in the opinion that the function point value adjustment factor is inadequate in its current form in adding value toward cost or effort estimating (1995, 13). When using function points as an input, the COCOMO II model uses <u>unadjusted</u> function points, converts them to equivalent lines of code (using <u>average</u> function point to lines of code conversion factors published by one author (Jones 1991, 76)), and proceeds from there with the cost estimating algorithm without considering the function point value adjustment factor at all. It is important to note, however, that COCOMO II is essentially a lines-of-code based cost estimating technique which can be adapted to accommodate function points, rather than a function points based technique. It is therefore subject to the shortcomings of the lines-of-code size metric, discussed in the literature.

A sampling of data from the database established by the International Software Benchmarking Standards Group (ISBSG) in fact indicates that much of the time the value adjustment factor as currently calculated is close to 1.0.

Limitations of the value adjustment factor as currently calculated from the values of the 14 GSCs include:

1. The 14 GSCs are equally weighted.
2. Their influence is additive (linear).
3. Each of the system characteristics reflected by the 14 GSCs can only contribute a maximum of 5% variation in the adjusted function point count from the unadjusted count. (Boehm notes that this is not consistent with the experience of the developers of COCOMO and COCOMO II (1997, 25-26).)

What Is Not Being Done

While there is significant criticism in the literature of the current method for determining the value adjustment factor, nowhere is there a proposed alternative which addresses the limitations cited above. To be sure, there are variations on the overall function points approach (Jones' Feature Points (Jones 1991, 81-94), Symons' Function Points Mark II (Symons 1988), and Whitmire's 3D Function Points (Whitmire 1995)), but these would render obsolete the accumulated body of function point data.

A different method for computing the value adjustment factor is needed, one which will address the existing limitations but which will permit the continued use of already collected data, toward the objective of making adjusted function points correlate more closely with development effort and cost. The intent of seeking such a method, then, is to find a better, more effective way to use existing data. If such a technique can be identified, it will represent a significant contribution toward making function point analysis a useful tool for the software development manager in estimating effort/costs, planning the allocation of resources, and forecasting workload.

The treatment of software development cost drivers by COCOMO II suggests such an approach.

A Proposed Alternate Approach

The function point sizing model uses the following approach to calculate the effects of the software complexity reflected in the assessments of the complexity levels of the 14 GSCs (Garmus and Herron 1996a, 92):

$$\text{AFP} = \text{UFP} \times \text{VAF} \qquad (3.1)$$

where AFP = adjusted function points, UFP = unadjusted function points, and VAF = value adjustment factor. The value adjustment factor is calculated using the additive model as follows (Garmus and Herron 1996a, 90):

$$\text{VAF} = 0.65 + 0.01\sum_{i=1}^{14}\text{GSC}_i \qquad (3.2)$$

where each GSC_i may take an integer value of 0 through 5 (Dreger 1989, 66). This results in the equation:

$$\text{AFP} = \text{UFP} \times \left[0.65 + 0.01\sum_{i=1}^{14}\text{GSC}_i\right] \qquad (3.3)$$

The COCOMO II approach, on the other hand, uses the following basic relationship (Boehm 1997, 13):

$$\text{Effort} = \text{Constant} \times \text{SLOC} \times \prod_{i=1}^{17}\text{CD}_i \qquad (3.4)$$

or, rewriting:

$$\text{Effort} = \text{Constant}\ [\text{UFP} \times (\text{Language-Dependent FP-to-LOC Conversion Factor}) \times \prod_{i=1}^{17}\text{CD}_i] \qquad (3.5)$$

where CD_i are the 17 COCOMO II "cost drivers" or "effort multipliers." These are

1. Multiplicative instead of additive.

2. Not necessarily of equal value.

3. Not constrained in the values they can take.

The 14 general system characteristics used in function point analysis are:

Data Communications	On-Line Update
Distributed Data Processing	Complex Processing
Performance	Reusability
Heavily Used Configuration	Installation Ease
Transaction Rate	Operational Ease
On-Line Data Entry	Multiple Sites
End-User Efficiency	Facilitate Change

Each of these is described briefly in Table 1 (excerpted from Jones (1991, 64-67)).

Table 1

Function Point General System Characteristics

GSC #	Name	Description
1	Data Communications	Data communication implies that data and/or control information would be sent or received over communication facilities.
2	Distributed Data Processing	Distributed functions are concerned with whether an application is monolithic and operates on a single contiguous processor or is distributed among a variety of processors.
3	Performance	Performance objectives are scored as 0 if no special performance criteria are stated by the users of the application and scored as 5 if the users insist on very stringent performance targets that require considerable effort to achieve.
4	Heavily Used Configuration	Heavily used configuration is scored as 0 if the application has no special usage constraints and as 5 if anticipated usage requires special effort to achieve.

Table 1 Continued

GSC #	Name	Description
5	Transaction Rate	Transaction rate is scored 0 if the volume of transactions is not significant and 5 if the volume of transactions is high enough to stress the application and require special effort to achieve desired throughputs.
6	On-Line Data Entry	On-line data entry is scored 0 if none or fewer than 15 percent of the transactions are interactive and 5 if all or more than 50 percent of the transactions are interactive.
7	End-User Efficiency	Design for end-user efficiency is scored 0 if there are no end users or if there are no special requirements for end users and 5 if the stated requirements for end-user efficiency are stringent enough to require special effort to achieve them.
8	On-Line Update	On-line update is scored 0 if there is none and 5 if on-line updates are both mandatory and especially difficult, perhaps because of the need to back up or protect data against accidental change.
9	Complex Processing	Complex processing is scored 0 if there is none and 5 in cases requiring extensive logical decisions, complicated mathematics, tricky exception processing, or elaborate security schemes.
10	Reusability	Reusability is scored 0 if the functionality is planned to stay local to the current application and 5 if much of the functionality and the project deliverables are intended for widespread utilization by other applications.
11	Installation Ease	Installation ease is scored 0 if this factor is insignificant and 5 if installation is both important and so stringent that it requires special effort to accomplish a satisfactory installation.
12	Operational Ease	Operational ease is scored 0 if this factor is insignificant and 5 if operational ease of use is so important that it requires special effort to achieve it.
13	Multiple Sites	Multiple sites is scored 0 if there is only one planned using location and 5 if the project and its deliverables are intended for many diverse locations.

Table 1 Continued

GSC #	Name	Description
14	Facilitate Change	Facilitate change is scored 0 if change does not occur, and 5 if the application is developed specifically to allow end users to make rapid changes to control data or tables which they maintain with the aid of the application.

The COCOMO II Cost Drivers are grouped into four categories:

Product Factors

Platform Factors

Personnel Factors

Project Factors (Development Environment)

The 17 COCOMO II Cost Drivers are described in Table 2 (excerpted from Boehm (1997, 35-43)).

Table 2

COCOMO II Cost Drivers

COCOMO II Designation	Name	Description
Product Factors:		
RELY	Required Software Reliability	This is the measure of the extent to which the software must perform its intended function over a period of time. If the effect of a software failure is only slight inconvenience, then RELY is low. If a failure would risk human life, then RELY is very high.

Table 2 Continued

COCOMO II Designation	Name	Description
DATA	Data Base Size	This measure attempts to capture the effect large data requirements have on product development. The rating is determined by calculating the ratio of database size in bytes to program size in SLOC. The reason the size of the database is important to consider is because of the effort required to generate the test data that will be used to exercise the program.
CPLX	Product Complexity	The complexity rating is the subjected weighted average of the following five areas: control operations, computational operations, device-dependent operations, data management operations, and user interface management operations.
RUSE	Required Reusability	This factor accounts for the additional effort needed to construct components intended for reuse on the current or future projects. This effort is consumed with creating more generic design of software, more elaborate documentation, and more extensive testing to ensure components are ready for use in other applications.
DOCU	Documentation Match to Life-Cycle Needs	The rating scale for this factor is evaluated in terms of the suitability of the project's documentation to its life cycle needs. The scale ranges from very low (many life-cycle needs uncovered) to very high (very excessive for life-cycle needs).
Platform Factors:		
TIME	Execution Time Constraint	This is a measure of the execution time constraint imposed upon a software system. The rating is expressed in terms of the percentage of available execution time expected to be used by the system or subsystem consuming the execution time resource, and can range from nominal (less than 50%) to extra high (95%).
STOR	Main Storage Constraint	This rating represents the degree of main storage constraint imposed on a software system or subsystem. In spite of the remarkable increase in available processor execution time and main storage, many applications continue to expand to consume whatever resources are available, making this factor still relevant.

Table 2 Continued

COCOMO II Designation	Name	Description
PVOL	Platform Volatility	"Platform" is used here to mean the complex of hardware and software which the software product being developed calls on to perform its tasks. The rating ranges from low, where there is a major change every 12 months or longer, to very high, where there is a major change every two weeks.
Personnel Factors:		
ACAP	Analyst Capability	Analysts are personnel that work on requirements, high level design, and detailed design. The major attributes reflected in this rating are analysis and design ability, efficiency and thoroughness, and the ability to communicate and cooperate.
PCAP	Programmer Capability	The major factors considered in this rating are ability, efficiency and thoroughness, and the ability to communicate and cooperate. The evaluation should be based on the capability of the programmers as a team rather than as individuals.
AEXP	Applications Experience	This rating is dependent on the level of applications experience of the project team developing the software system or subsystem. The ratings are defined in terms of the project team's equivalent level of experience (from less than two months to more than six years) with the specific type of application.
PEXP	Platform Experience	This rating reflects the importance of understanding the use of more powerful platforms, including more graphic user interface, database, networking, and distributed middleware capabilities.
LTEX	Language and Tool Experience	This is a measure of the level of programming language and software tool experience of the project team developing the software system or subsystem. Software development includes the use of tools that perform requirements and design representation and analysis, configuration management, document extraction, library management, program style and formatting, consistency checking, etc.
PCON	Personnel Continuity	This rating is expressed in terms of the project's annual personnel turnover.
Environmental Factors:		
TOOL	Use of Software Tools	The software tool rating ranges from simple edit and code (very low) to integrated lifecycle management tools (very high).

Table 2 Continued

COCOMO II Designation	Name	Description
SITE	Multisite Development	Determining this rating involves the assessment and averaging of two factors: site collocation (from fully collocated to international distribution) and communication support (from surface mail and some phone access to full interactive multimedia).
SCED	Required Development Schedule	This rating measures the schedule constraint imposed on the project team developing the software, in terms of the percentage of schedule stretch-out or acceleration with respect to a nominal schedule for a project requiring a given amount of effort. A schedule compress of 74% is rated very low; a stretch-out of 160% is rated very high.

The 14 function point GSCs do not address personnel or environmental considerations, since these factors are not intrinsic to the software system itself. However, because they address system characteristics, with the system comprised of the product and the platform, they do relate to certain of the COCOMO II Product and Platform factors, specifically CPLX, TIME, and RUSE, as indicated in Table 3.

Clearly there are product and platform factors addressed in the COCOMO II Model which have no counterparts in the function point GSCs. This may be because of changes in the software development environment during the period 1984 (when the function point GSCs were defined) and 1995 (when COCOMO II was introduced); it may be because

Table 3

Relationship between Function Point GSCs
and COCOMO II Cost Drivers

Function Point GSC	Related COCOMO II Factor
1. Data Communications	CPLX
2. Distributed Data Processing	CPLX
3. Performance	TIME
4. Heavily Used Configuration	TIME
5. Transaction Rate	TIME
6. On-Line Data Entry	CPLX
7. End-User Efficiency	CPLX
8. On-Line Update	CPLX
9. Complex Processing	CPLX
10. Reusability	RUSE
11. Installation Ease	CPLX
12. Operational Ease	CPLX
13. Multiple Sites	CPLX
14. Facilitate Change	CPLX

COCOMO and COCOMO II were designed to be cost estimating models, whereas the function points approach, while producing an output clearly related to cost, was intended to provide an alternate metric (to SLOC) of software project size; or it may have had to do with the different perspectives of the respective developers of function points and COCOMO II, Allan Albrecht and Barry Boehm. That, however, is not the issue here. What is of interest is the fact that both approaches attempt to quantify (i.e., assign numerical values to) qualitative assessments (based on expert judgment) of the same general type of software system characteristics. However, as has been discussed, the treatment of these numerical assessments is substantially different between the two approaches.

If indeed the two approaches assess and quantify the same type of characteristics, then the numerical values so obtained should relate in somewhat the same manner to software project size and, ultimately, development cost. Since, according to many, the current method in the function points approach of applying the effects on size (and cost) of the GSCs is inadequate, the approach used in applying the 17 cost drivers in the COCOMO II Model may, if used in converting unadjusted to adjusted function point counts, offer the possibility of improving the usefulness of the function point GSCs and the resultant value adjustment factor.

Even though COCOMO II remains basically a lines-of-code based technique, the approach does suggest a different relationship by which to accommodate system complexity and other considerations, one that would alleviate certain of the existing concerns with calculation of the function point value adjustment factor using current procedures.

What is proposed, then, is to develop and evaluate an approach to apply the existing 14 GSCs in a different manner, one that is similar to that used by COCOMO II in applying its cost driver factors (i.e., multiplicative versus additive).

This has appeal in that it may allow the GSC data already incorporated in function point counting procedures and already collected over the past ten to fifteen years to be used to better advantage (i.e., to produce adjusted function point counts more closely correlated with development effort). Admittedly, personnel and environmental considerations still will not be addressed, but the proposed approach holds promise for accommodating them in the future.

The Research Question

The research question to be addressed herein, then, can be stated as follows: is it possible, using a relationship suggested by the COCOMO II Model, to develop an alternative means of determining the function point value adjustment factor which will result in a final or adjusted function point count which will correlate significantly better (than is currently the case using existing procedures) with software development effort and therefore cost? Restating this in the form of a null hypothesis: the correlation between the adjusted function point count and software development effort will not improve as a result of using an alternative means of determining the value adjustment factor suggested by COCOMO II cost driver relationships.

Expected Findings

It was expected that the proposed "new" method of calculating the adjusted function point value from unadjusted function points would produce results which are more closely correlated with actual development effort than are results obtaining using the existing method. Not only is a new form for assessing the influence of the 14 GSCs proposed, the calculated contributions of these 14 factors are not constrained as is currently the case.

When sufficient data are available to conduct a comparison of performance of the proposed model across different software application domains, it will be interesting to see if the model performs better for some application domains than for others, and to compare any such difference to that experienced using the current method. One of the criticisms of the function point approach is that it does not perform well in other than management information system (MIS) (business) applications. By removing some of the constraints on the degree to which individual GSCs can influence the value of the adjusted function point count, it is possible that the influence of certain of the characteristics of non-MIS applications on development effort can be more completely or accurately captured in the adjusted function point count.

Contributions

The research effort described herein addresses limitations of the function point approach in a manner not reflected in the literature to date. Although the literature addresses concerns with the usefulness of the value adjustment factor, which is based on the assessed values of the 14 GSCs, the research contributes by proposing a new treatment of the 14 GSCs and a new way of calculating the value adjustment factor which eliminates the basis for at least some of those concerns, those related to the manner in which the influence of the GSCs is captured (as discussed earlier). It is important to note that the proposed approach seeks to retain existing data elements and simply to modify the treatment of them, ensuring that already-collected data will still be usable and that current function point counting practices are still valid.

If it could be demonstrated that the results of the proposed research in fact do provide a new method of calculating the value adjustment factor which produces adjusted function point counts which are more closely correlated with actual development effort, then obtaining a valid estimate of development effort based on function points would be more straightforward once the adjusted function point count had been determined. Cost estimating using function points would therefore be easier for software development managers.

Furthermore, since no "rescaling" of factors reflecting the influence of the existing 14 GSCs would be necessary, it will be relatively simple to incorporate additional cost/effort "drivers" into the model. Most notably, such factors might include personnel and environmental factors, as suggested by the COCOMO II model.

Regardless of the outcome (whether an "improvement" is realized in the value adjustment factor), the results of this research contribute to a deeper understanding of the key issues involved in relating software size to development costs and, in particular, the manner in which cost drivers other than size (as the most significant single cost driver) modulate the effects of software size in development cost prediction. This research also contributes to the industry (the management of software development), by tending either to offer or eliminate an alternative approach which has the potential for improving the correlation of the adjusted function point count with software development effort and cost. And, the results of this research make a contribution to the literature pertaining to function points and to software cost estimating literature in general, as well as opening new avenues and directions for future research.

The results of the research also underscore the need for COCOMO II to be able to accommodate function points directly as an input. This could be done by modifying the COCOMO II model or by developing a function points variant to the model. Currently, in order to use function points with COCOMO II, one must apply an "industry average" conversion factor to produce an ostensibly equivalent lines-of-code figure from the function point count for input into the COCOMO II model.

Summarizing the contributions of the analysis and discussion contained herein: first, this research demonstrates the application of a different treatment of existing parameters (GSCs) used to adjust a software size metric, one which removes certain constraints on the effect these parameters may have. This new approach responds to criticism which has been levied against the current method. Based on the results of applying the new approach to data from one set of software development projects, there is a strong indication that the parameters themselves (i.e., the GSCs) may be inadequate for what they purport to do, namely capture the effects of "system characteristics" in adjusting the value of the size metric, the purpose of the metric being to compare projects, measure productivity, and support effort and cost estimation and prediction. Additionally, an indication is provided of those individual system characteristics (GSCs) identified in the function point analysis process which do significantly affect development and cost as well as those which have little or no effect. The research provides the groundwork for investigation into the mechanics by which variations in "system characteristics" impact software development effort and cost. Directions are offered for further research which is indicated by the work here, specifically assessment of the proposed new approach using a better set of data (i.e., a better cross-section of modern software project data) and investigation into the adequacy and appropriateness of the 14 existing GSCs. Contributions to the practice and methodology of software cost estimating include: (1) demonstration of the feasibility of an alternative approach; (2) identification of the need to re-evaluate the appropriateness of the existing GSCs; and (3) identification of a need for better dialogue between IFPUG and the COCOMO II proponents, with a "closing of the ranks" so that function points become a direct input to the COCOMO II cost estimating model. The current need in COCOMO II to convert function points to equivalent lines-of-code using industry average conversion factors only introduces another source of error or inaccuracies.

CHAPTER IV

RESEARCH METHODOLOGY

Development of a Proposed New Model for Determining Adjusted Function Points

The first step in accomplishing the proposed research was as follows: the "provisional values" (Boehm 1997, 73) of the multipliers corresponding to each of the qualitative ratings assigned to each of the 17 cost driver factors of COCOMO II were studied. It was observed that in each case the value of the multiplier corresponding to the "nominal" rating is 1.0. Boehm explains that "the average effort multiplier [weight] assigned to a cost driver is 1.0, and the rating level associated with that weight is called Nominal" (1997, 13). The following exponential form (Equation (4.1)) will consistently produce a value of 1.0 when $k = 0$, regardless of the value of n:

$$\text{Value} = (n)^{k} \tag{4.1}$$

From examination of the values of the COCOMO II multipliers, n was found to be a number between 1.00 and 1.25 when a higher rating for the cost driver results in a higher cost estimate and between 0.80 and 1.00 when a higher rating for the cost driver results in a lower cost estimate.

In order to determine if values of *n* could be identified which would produce reasonable approximations of the COCOMO II values for the multipliers when *k* was a number other than zero, the following approach was used. First, sequential positive integer values were assigned to values of *k* for COCOMO II cost driver factor rating levels above "nominal" and sequential negative integer values were assigned for rating levels below "nominal." Sequential integers were chosen because values assigned the possible rating levels for the function point GSCs also consist of sequential integers ranging from 0 through 5 (Dreger 1989, 66). The significance of this will be demonstrated later in the discussion.

An approach suggested by Conte, Dunsmore, and Shen (1986, 172-173) was then applied. For each of the 17 cost drivers, different values of *n*, rounded to two decimal places for consistency with the COCOMO II provisional values, were used to calculate the value of the multiplier for the various non-zero values of *k*. The magnitude of the relative error (MRE), defined as

$$\text{MRE} = \left| \frac{M_n - M_c}{M_c} \right| \tag{4.2}$$

was determined for each rating level (*k*), where M_n is the calculated value of the multiplier using the exponential relationship (Equation (4.1)) and M_c is the value of the multiplier obtained from the COCOMO II Model Definition Manual (Boehm 1997, 73). The "error" in this case, for a given value of *k*, is the difference between the calculated value of the multiplier using Equation (4.1) and the COCOMO II value of the multiplier. For each cost driver, the value of *n* which resulted in the lowest mean MRE for the non-zero values of *k* was therefore the one which provided calculated multiplier values which best approximated the COCOMO II multiplier values.

The results of this process are illustrated in Table 4. For each of the 17 cost drivers, the 1997 "provisional" value of the multiplier, based on analysis of 83 software development projects (Boehm 1997), is listed for each applicable *k* value, followed by the results obtained using Equation (4.1), with the value of *n* determined as described above.

Table 4

Comparison of COCOMO II Cost Driver Provisional Values with Corresponding Values Obtained Using Proposed Exponential Form

		Rating Level					
Cost Driver		Very Low $k = -2$	Low $k = -1$	Nominal $k = 0$	High $k = +1$	Very High $k = +2$	Extra High $k = +3$
RELY	COCOMO II	0.75	0.88	1.00	1.15	1.39	
	$n = 1.16$	0.74	0.86	1.00	1.16	1.35	
DATA	COCOMO II		0.93	1.00	1.09	1.19	
	$n = 1.09$		0.92	1.00	1.09	1.19	
CPLX	COCOMO II	0.75	0.88	1.00	1.15	1.30	1.66
	$n = 1.15$	0.76	0.87	1.00	1.15	1.32	1.52
RUSE	COCOMO II		0.91	1.00	1.14	1.29	1.49
	$n = 1.14$		0.88	1.00	1.14	1.30	1.48

Table 4 Continued

		Rating Level					
Cost Driver		Very Low $k = -2$	Low $k = -1$	Nominal $k = 0$	High $k = +1$	Very High $k = +2$	Extra High $k = +3$
DOCU	COCOMO II $n = 1.06$	0.89 0.89	0.95 0.94	1.00 1.00	1.06 1.06	1.13 1.12	
TIME	COCOMO II $n = 1.15$			1.00 1.00	1.11 1.15	1.31 1.32	1.67 1.52
STOR	COCOMO II $n = 1.10$			1.00 1.00	1.06 1.10	1.21 1.21	1.57 1.33
PVOL	COCOMO II $n = 1.15$		0.87 0.87	1.00 1.00	1.15 1.15	1.30 1.32	
(For the following cost drivers, a lower rating implies a higher multiplier; consequently, the "n" values are < 1.0.)							
ACAP	COCOMO II $n = 0.82$	1.50 1.49	1.22 1.22	1.00 1.00	0.83 0.82	0.67 0.67	
PCAP	COCOMO II $n = 0.86$	1.37 1.35	1.16 1.16	1.00 1.00	0.87 0.86	0.74 0.74	
PCON	COCOMO II $n = 0.91$	1.24 1.21	1.10 1.10	1.00 1.00	0.92 0.91	0.84 0.83	
AEXP	COCOMO II $n = 0.90$	1.22 1.23	1.10 1.11	1.00 1.00	0.89 0.90	0.81 0.81	
PEXP	COCOMO II $n = 0.89$	1.25 1.26	1.12 1.12	1.00 1.00	0.88 0.89	0.81 0.79	
LTEX	COCOMO II $n = 0.91$	1.22 1.21	1.10 1.10	1.00 1.00	0.91 0.91	0.84 0.83	
TOOL	COCOMO II $n = 0.89$	1.24 1.26	1.12 1.12	1.00 1.00	0.86 0.89	0.72 0.79	
SITE	COCOMO II $n = 0.92$	1.25 1.18	1.10 1.09	1.00 1.00	0.92 0.92	0.84 0.85	0.78 0.78
SCED	COCOMO II $n = 0.91$	1.29 1.21	1.10 1.10	1.00 1.00	1.00 0.91	1.00 0.83	

Additionally, these comparisons are displayed graphically in Appendix A.

It was found that in most cases there was a very close "fit" between the value of the COCOMO II multiplier and the multiplier value which resulted from Equation (4.1) for the value of *n* determined as described above. This was observed to be true for values of *k* over the range of interest (k = -2 to k = +3). Larger values of *k* (positive and negative) were not considered as they would have no counterpart in a COCOMO II rating level. The results presented in Table 4 and Appendix A were sufficiently good over the range of interest to warrant investigating if such a relationship can be applied to advantage in seeking a better way to calculate the function point value adjustment factor.

The next step, therefore, was to devise a means whereby the six possible ratings of each function point GSC can be assigned values which will permit use of the relationship (Value = $(n)^k$) in calculating the value adjustment factor. According to the literature pertaining to the counting of function points (Dreger 1989, 66; Jones 1991, 65; Garmus and Herron 1996a, 81), a value of 3 is assigned to a GSC if it has "average" influence on the particular software effort being addressed. If the six possible numerical values each GSC may take are "remapped" onto a scale of -3 through 2, as follows:

Old	New
0	-3
1	-2
2	-1
3	0
4	1
5	2

then the value when the GSC has "average" influence becomes 0 vice 3, corresponding to the *k* value of 0 for the "nominal" rating in COCOMO II.

For a suitable alternative method of calculating adjusted function points (AFP) from unadjusted function points (UFP), then, the following form is proposed (Equation (4.3) is identical to Equation (3.1)):

$$\mathrm{AFP} = \mathrm{UFP} \times \mathrm{VAF} \tag{4.3}$$

$$\mathrm{VAF} = \prod_{i=1}^{14} n_i^{k_i} \tag{4.4}$$

Therefore:

$$\mathrm{AFP} = \mathrm{UFP} \times \prod_{i=1}^{14} n_i^{k_i} \tag{4.5}$$

where k_i is the rating value (integers -3 through 2) of the ith GSC, and the n_i are numbers to be determined.

The Current Research Effort with Respect to Software Cost Estimating Theory and Practice

It is important at this point to place the current research effort in perspective relative to the theory and practice of software cost estimating.

The body of theory pertaining to the software cost estimating process is quite small and is incomplete. Conte, Dunsmore, and Shen (1986, 274) cite the "lack of progress in scientific approaches to this problem." They note that "there are literally hundreds of factors that may affect productivity and hence effort" (280). They also note, however, that many of these are insignificant and can be ignored, and that others are highly correlated and can be combined into a single factor. The implication here is that it is important to attempt to identify and capture the effects of significant, substantially independent factors which affect effort and hence cost.

Conte, Dunsmore, and Shen (1986, Chapter 6) identify and discuss in detail four categories of software cost estimation models: historical-experiential, statistically-based, theoretically-based, and composite.

Historical-experiential models may be simplistically described as expert judgment, either individual or collective. Statistically-based models are subdivided into linear statistical models and nonlinear statistical models.

Linear statistical models are generally of the form

$$\text{Effort} = c_0 + \sum_{i=1}^{n} c_i x_i \qquad (4.6)$$

where the x_i are software attributes or factors that are believed to affect software development effort (sometimes called cost driver attributes).

According to Conte, Dunsmore, and Shen (281), most of the nonlinear statistical models are of the form

$$\text{Effort} = (a + bS^{c})\, m(X) \qquad (4.7)$$

where:

> *S* is the size of the project;
>
> *a*, *b*, and *c* are constants usually derived by regression analysis; and
>
> *m*(*X*) is an adjustment multiplier that depends on one or more cost driver attributes.

Conte, Dunsmore, and Shen also point out that, in some cases, *a*, *b*, and *c* may also be functions of one or more cost drivers and advise that *m*(*X*) can be a complicated function of several variables (281-282). The need for large amounts of data, consistently defined, for empirically determining the values of the multipliers and constants becomes obvious and provides an indication of how data-constrained the practical implementation of such models can become. If $a = 0$ and $m(X) = 1$, it can be seen that Equation (4.7) is the equivalent of Equation (1.1) (Wellman, 1992, 36).

A third category of cost estimation model cited by Conte, Dunsmore, and Shen is that of theoretically-based models. They advise that such models are "based on theories of how the human mind functions during the programming process and on mathematical laws that the software process is assumed to follow." They describe three such theoretically-based models in detail. Two of these are of a form which makes it appropriate to mention them with respect to the current research. These are the Putnam Resource Allocation Model and the Jensen Model. The Putnam Model can be written as (Conte, Dunsmore and Shen, 294):

$$\text{Effort} = \frac{CS^3}{T^4} \tag{4.8}$$

where *S* is project size in terms of lines of code produced, *T* is development time, and *C* is described as a "technology constant." Relating these terms to the cost drivers which have been discussed herein with respect to the function point value adjustment factor and the COCOMO II cost model: first, the inverse fourth power relationship of time with effort would equate to a direct such relationship of effort with the COCOMO II schedule constraint factor (SCED), an environmental factor, or "characteristic," not captured by the function point GSCs. Putnam's work apparently led him to the conclusion that development time is a major factor affecting effort; however, Conte, Dunsmore, and Shen (291) point out that other researchers do not support the severe penalty imposed on schedule constraints by Putnam's "fourth power" law. The technology constant *C* is intended to reflect the effect on productivity of such factors as hardware constraints, program complexity (these two constitute "system" characteristics), personnel experience levels, and the programming environment (in other words, personnel and environmental factors). According to Conte, Dunsmore, and Shen (290), Putnam proposed either using a discrete spectrum of 20 predetermined values of *C* (which presumably he had developed) or using historical project data (which, of course, introduces an empirical element). Conte, Dunsmore, and Shen used a least squares regression to obtain values of *C* from historical data, then applied those values to "several databases." In their words, the results showed "uniformly poor performance."

The Jensen Model is similar to Putnam's. It can be written as follows (Conte, Dunsmore, and Shen, 296):

$$\text{Effort} = \frac{cS^2}{T^2} \tag{4.9}$$

where *S* and *T* represent the same parameters as described above for the Putnam Model and *c* is a constant which incorporates a "basic technology" factor as well as product, personnel, and computer (hardware) considerations, plus scale modifications. Conte, Dunsmore, and Shen conducted an evaluation of the Jensen Model similar to that applied to the Putnam Model, above (including the use of least squares regression with historical data to determine values for *c*), and found performance "slightly better than that of the Putnam Model, but ... still very poor."

The third theoretically-based model discussed by Conte, Dunsmore, and Shen (296-300) is the Software Science Effort Model. Its approach is substantially different from the Putnam and Jensen Models. The assumptions underlying the development of this model limit its theoretical basis to small, one-programmer projects. Conte, Dunsmore, and Shen point out that, although the "theory of Software Science initially captured the imaginations of many researchers since it proposed what appeared to be a sound theoretical basis for understanding the human mental processes involved in programming," that "subsequent research has cast considerable doubt on the psychological assumptions underlying the theory." They further advise that "the weight of empirical evidence ... tends to dispute the validity" of the model "as an effort estimator on more realistic projects."

Finally, Conte, Dunsmore, and Shen (300) introduce a fourth category of cost estimating model which they call composite models. Composite models incorporate a combination of analytic equations, statistical data fitting, and expert judgment. They cite COCOMO as "the best known of all composite models" (in 1986) and advise that COCOMO is the most complete and thoroughly documented of all models for effort estimation. COCOMO II is basically an update, some 16 years later, of the initial COCOMO model.

Conte, Dunsmore, and Shen (301) present the basic COCOMO equation as being of the form:

$$\text{Effort} = a_i S^{b_i} m(X) \tag{4.10}$$

where a_i varies with the mode (three modes are identified: organic, semidetached, and embedded) and level (basic, intermediate, and detailed, as identified earlier in Chapter I), and b_i varies only with the mode. The $m(X)$ term is a composite multiplier that depends on the values of 15 cost driver attributes.

In the update of COCOMO to COCOMO II, the form was modified to the following (Boehm 1997, 13):

$$\text{Effort} = A\, S^B\, m(X) \tag{4.11}$$

where A is an empirically determined constant and the exponent B is used to capture economies of scale. The COCOMO II Model Definition Manual (Boehm 1997, Chapter 3) provides a detailed discussion of how the economies of scale are assessed. If $B < 1.0$ for a given project, the project exhibits economies of scale; if $B > 1.0$, the project exhibits diseconomies of scale; and if $B = 1.0$, the economies and diseconomies of scale are in balance. In COCOMO II, the composite multiplier $m(X)$ reflects the effects of 17 cost drivers, as is discussed elsewhere herein. If the value of B is taken as 1.0, it can easily be seen that Equation (4.11) is the equivalent of Equation (3.4).

The discussion of the various categories of software cost estimation model by Conte, Dunsmore, and Shen leads to three conclusions: first, the body of theory which exists in support of software cost estimating is very limited; second, the theoretically-based models described by Conte, Dunsmore, and Shen place a heavy reliance on historical, or empirical, data; and finally, the majority of these models assume a multiplicative relationship between effort and cost drivers, technology factors, time constraints, etc. In other words, a multiplicative relationship has been assumed between effort and product and platform factors (system characteristics), as well as personnel and environmental factors.

In using lines-of-code as the size metric, even though the definition of a "line-of-code" is subject to some interpretation, and the term must therefore be precisely defined, one is working with a tangible entity. However, such is not the case with function points. Function points, as has been discussed, are a synthetic metric whose value lies in what they represent for purposes of correlation and comparison. As Abran and Robillard (1994, 181) note, "function points do not derive from a well-defined and proven theory; they are entirely empirically based on expert opinion." When Albrecht attempted to incorporate the effects of the general system characteristics into the final, or adjusted, function point count, in effect he was incorporating some portion of the effect of the cost driver adjustment multiplier, $m(X)$ in the preceding equations, into the size metric S (i.e., the adjusted function point count). The multiplicative relationship assumed in this case as opposed to an additive relationship for capturing such effects in the adjusted function point count is therefore consistent with most software cost estimating models found in the literature.

The exponential form, n^k, relationship used earlier for determining the values of multipliers is not addressed in the limited software cost estimating theory that exists. It originated as a result of the analysis described earlier. From this analysis, a computational means was provided for approximating the behavior of similar multipliers, which were based on historical data for 83 projects, developed for use in the COCOMO II cost estimating model by Boehm (1997, 73).

Determining and Applying Function Point Multipliers for the Revised Model

The next step was to determine the numbers (the n_i) to be used as the basis for calculating the function point multipliers using the exponential form shown in Equation (4.1) and the "remapped" rating levels of the GSCs, assigned numerical values of -3 through 2. Assuming a directly proportional relationship between development effort and the adjusted function point value (since attaining a closer such relationship is a stated goal of the research), this relationship may be stated as follows:

$$\text{Effort} = \text{Constant} \times \text{AFP} = \text{Constant} \times \text{UFP} \times \prod_{i=1}^{14} n_i^{k_i} \tag{4.12}$$

or, converting to logarithmic form:

$$\text{Log (Effort)} = \text{Log (Constant)} + \text{Log (UFP)} + \sum_{i=1}^{14} k_i \text{ Log } n_i \tag{4.13}$$

Rewriting the equation results in:

$$\text{Log (Effort)} - \text{Log (UFP)} = \text{Log (Constant)} + \sum_{i=1}^{14} (\text{Log } n_i) k_i \tag{4.14}$$

This produces an equation to which the following linear statistical model can be applied:

$$y = \alpha + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_i x_i + \varepsilon \tag{4.15}$$

where:

$$y = [\text{Log (Effort)} - \text{Log (UFP)}] \tag{4.16}$$

$$\alpha = \text{Log (Constant)} \tag{4.17}$$

$$\beta_i = \text{Log } n_i \tag{4.18}$$

$$x_i = k_i \tag{4.19}$$

The random error term, ε, would contain the effect of personnel and environmental cost drivers which it is already acknowledged are not being captured. It would also include the effects of any "system characteristic" cost drivers not captured by the 14 currently defined

GSCs, as well as any other variation between theoretical values and empirical results.

Given that sufficient project data are available or can be collected, by applying regression analysis to a random sampling (i.e., a portion) of those data, values b_i which are estimates of β_i (or Log n_i) can be determined. From there estimated values for the n_i can be determined.

With these values of n_i, the "new" adjusted function point counts for the remainder of the data can be calculated, using

$$\text{AFP} = \text{UFP} \times \prod_{i=1}^{14} n_i^{k_i} \tag{4.5}$$

as well as the "traditional" adjusted function point counts using Equation (3.3), restated here as Equation (4.20):

$$\text{AFP} = \text{UFP} \times \left[0.65 + 0.01 \sum_{i=1}^{14} \text{GSC}_i \right] \tag{4.20}$$

A statistical comparison can then be conducted to determine if AFP_{new} correlates significantly better with actual development effort than does AFP_{old}.

Data Collection and Methodology

Obtaining adequate data with which to accomplish the research proved to be much more difficult than anticipated. With an earlier version of the planned research approach, one that required individuals to respond to a survey, 30 members of IFPUG, as a pilot group to "test the waters," were sent survey packages and questionnaires by mail. There were no responses. Next, an appeal was broadcast over the Internet to the approximately 800 members of the Function Point "ListServ" Group (the ListServ is an Internet forum of function point practitioners, researchers, and other interested individuals, moderated at the Software Engineering Management Research Laboratory at the University of Quebec at Montreal). While there were a few responses, these resulted in no data. A second ListServ appeal yielded data from one individual for two projects. These are included in the data for the research reported herein.

The research plan was modified to be less ambitious in its data requirements, and another request was made to the ListServ moderator, Denis St-Pierre of the University of Quebec at Montreal. St-Pierre referred this researcher to Jean-Marc Desharnais, also of the University of Quebec at Montreal, who graciously provided the data for 299 software development projects. These data had been collected by Desharnais for projects completed over the period 1982 through 1990. Data for a portion of these projects had been used by Desharnais in his Masters thesis (Desharnais 1988), and the data for all 299 projects were used as the basis for the analysis reported by Finnie, Wittig, and Desharnais (1997). Data for two projects, both completed in 1997, were provided by

Dillard Boland of Computer Sciences Corporation, in response to an earlier appeal to the Function Point ListServ Group.

Relevant data for the 301 projects, 299 supplied by Desharnais and two provided by Boland, were transferred into a Microsoft Excel Version 5.0 PC-based computer file. A printout of these data is provided as Appendix B. A description of the data elements in Appendix B is provided here:

Project No: A sequential project number. The 299 projects for which data were provided by Desharnais are numbered 1 through 299.

Effort (mh): Software development effort in manhours.

UFP: The total unadjusted function points for the project.

GSC #1 - GSC #14: Assessed values for the 14 general system characteristics for the project.

AFP_{old}: The adjusted function point value calculated using the current method (see Equation (4.20)).

Approximately 70 percent of the projects (210) were randomly selected from the total of 301, using the PC-based Random Number Generator Program written by Graziano and Raulin of the State University of New York at Buffalo. These were entered into a separate Microsoft Excel file. The appropriate conversions of data elements for these 210 projects were made as necessary to support the regression analysis discussed above. These data are displayed in Appendix C. A description of the data elements in Appendix C, not already discussed earlier, is provided here:

> GSC(adj)1 - GSC(adj)14: Assessed values for the 14 general system characteristics for the project adjusted so that a value of zero corresponds to an assessment of "average."
>
> LOG(Effort): Logarithm (to the base 10) of the number of manhours of effort.
>
> LOG(UFP): Logarithm (to the base 10) of the number of unadjusted function points.

Estimated values for the n_i for the 14 GSCs, following the approach described earlier, were then calculated.

Using these values for the n_i in the relationship

$$\mathrm{AFP} = \mathrm{UFP} \times \prod_{i=1}^{14} n_i^{k_i} \qquad (4.5)$$

adjusted function point values, using the proposed new approach, were calculated for the remaining 91 projects. Data for these projects, including the $\mathrm{AFP}_{\mathrm{new}}$ value for each project, were entered into a Microsoft Excel file and are displayed in Appendix D. As before, a description of the data elements in Appendix D, not already discussed, is provided here:

> Overall Multiplier: The value adjustment factor calculated using the proposed new approach (see Equation (4.4)).
>
> AFPnew: The adjusted function point value calculated using the proposed new approach (see Equation (4.5)).

Next, sample correlation coefficients for the 91 projects were determined for the correlation of AFP_{old} with development effort and for the correlation of AFP_{new} with development effort.

The mathematical computations and the statistical analysis were accomplished using the Function and Data Analysis Tools, respectively, contained in Microsoft Excel Version 5.0. Stepwise regression was accomplished using the JMP statistical software, Version 3.2.2, manufactured by SAS Institute Inc.

Finally, testing of the significance in the difference between the two correlation coefficients, as described in the following chapter, was conducted.

A summary listing of the steps involved in the data collection and methodology is provided below:

1. Identify data requirements based on proposed research approach.
2. Solicit data from likely sources.
3. Obtain data and consolidate in a PC-based spreadsheet file (Microsoft Excel Version 5.0)
4. Use a random number generator to randomly select approximately 70 percent of the projects (210 in this case) for use in developing n_i values for the model.
5. Enter these projects into a separate spreadsheet file.
6. Use the steps outlined in Equations (4.4), (4.5), and (4.12) through (4.19) and the accompanying discussion to calculate values for the n_i.

7. Enter the remaining 30 percent of the projects (91) into a separate spreadsheet file.

8. Calculate the AFP_{new} values for these 91 projects using the proposed new approach and using the n_i values calculated in Step 6 above.

9. Determine the sample correlation coefficient for the correlation of AFP_{old} with development effort and for the correlation of AFP_{new} with development effort.

10. Test for significance in the difference between the two correlation coefficients.

CHAPTER V

RESULTS AND CONCLUSIONS

Results

Linear regression was applied to the data in the logarithmic form as described in the previous chapter. The multiple correlation coefficient, R^2, was 0.24, indicating that only 24 percent of the variability (in the logarithmic form upon which the regression was applied) was accounted for by the resultant regression equation. While this was of some concern, it has already been acknowledged that certain software development cost driver factors are not captured by the 14 GSCs. The contribution made by each of the 14 GSCs can be seen in Table 5, which provides the results of stepwise regression.

The n_i values obtained by the regression analysis, for use in calculating adjusted function point values from the relationship, shown earlier as Equation (4.5)

$$\mathrm{AFP} = \mathrm{UFP} \times \prod_{i=1}^{14} n_i^{k_i} \tag{5.1}$$

are provided in Table 6.

It is of interest to note that, based on project data from the 210 projects, the values of the n_i for three of the GSCs (1, 5, and 6) are less than one, meaning that the higher the rating for the presence of each of

Table 5

Results of Stepwise Regression

GSC Number	GSC Description	Coefficient	*t*-statistic	*F* Ratio	Prob. *F*
9	Complex Processing	0.05226	3.7889	14.356	0.0002
13	Multiple Sites	0.05056	3.3002	10.891	0.0011
6	On-Line Data Entry	-0.01779	-1.3061	1.706	0.1931
14	Facilitate Change	0.01898	1.3721	1.883	0.1716
1	Data Communi-cations	-0.02233	-1.5348	2.356	0.1264
8	On-Line Update	0.02089	1.1689	1.366	0.2439
3	Performance	0.01611	1.0511	1.105	0.2945
5	Transaction Rate	-0.02474	-1.3787	1.901	0.1696
10	Reusability	0.01461	0.8717	0.760	0.3844
12	Operational Ease	0.01119	0.6565	0.431	0.5123
4	Heavily Used Configuration	0.00511	0.2792	0.078	0.7804
11	Installation Ease	-0.00127	-0.0845	0.007	0.9327
2	Distributed Data Processing	0.00045	0.0223	0.000	0.9822
7	End-User Efficiency	0.00007	0.0040	0.000	0.9968

Table 6

Proposed Function Point Multipliers (n_i)

GSC No.	Description	n_i	GSC No.	Description	n_i
1	Data Communications	0.95	8	On-Line Update	1.05
2	Distributed Data Processing	1.00	9	Complex Processing	1.13
3	Performance	1.04	10	Reusability	1.03
4	Heavily Used Configuration	1.01	11	Installation Ease	1.00
5	Transaction Rate	0.95	12	Operational Ease	1.03
6	On-Line Data Entry	0.96	13	Multiple Sites	1.12
7	End-User Efficiency	1.00	14	Facilitate Change	1.05

those particular characteristics, the lower the value of the value adjustment factor and the adjusted function point count using the proposed new approach. The direction of this effect is opposite that which results from current function point counting procedures. Further, three other of the GSCs (2, 7, and 11) had n_i values extremely close to 1.0, meaning that differences in the GSC rating value have little effect on the (revised) adjusted function point count.

Such an outcome (negative influence or no influence) was not anticipated for six of the 14 GSCs, as it seemed reasonable to expect an increasing positive contribution to the adjusted function point count as GSC rating values increased. In fact, for the sample of 210 projects, the regression coefficients of only two of the cost drivers (in the logarithmic equation), GSC number 9 (Complex Processing) and GSC number 13 (Multiple Sites), were significant at the five percent level (using the *t*-statistics for the individual coefficients). These two cost drivers (GSCs number 9 and 13) were also those with the highest resultant n_i value (see Table 6) and therefore those whose multipliers (resulting from rating levels other than "average") will have the greatest impact on the calculated value of the value adjustment factor using the proposed new approach.

Nonetheless, in order to proceed with development of the model, the proposed multiplicative relationship using all 14 of the n_i values was applied to data for the remaining 91 projects to determine values for AFP_{new}. As indicated earlier, these data, including calculated values for AFP_{new}, are displayed in Appendix D. Coefficients of correlation were then calculated for the sample for the correlation between AFP_{old} and development effort and for the correlation between AFP_{new} and development effort, with the following results:

$$r_{\text{AFP}_{\text{old}},\text{effort}} = 0.744502 \tag{5.2}$$

$$r_{\text{AFP}_{\text{new}},\text{effort}} = 0.703164 \tag{5.3}$$

Clearly this is contrary to the expected result: For the sample, the current method produced adjusted function point values more closely correlated to effort than did the proposed new method, in spite of the elimination of certain constraints with the new method. In testing for significance in the difference between the two correlation coefficients, the guidance provided by Downie and Heath (1974, 228) was followed. For a situation such as that which exists here, in which "variable 1 is correlated with variable 2, and variable 1 is also correlated with variable 3, and all measurements are made on the same sample," they suggest a t test with t calculated as follows:

$$t = \frac{(r_{12} - r_{13})\sqrt{(N-3)(1+r_{23})}}{\sqrt{2(1 - r_{12}^2 - r_{13}^2 - r_{23}^2 + 2r_{12}r_{13}r_{23})}} \tag{5.4}$$

This is interpreted by going into the *t* table with N - 3 degrees of freedom. In the case at hand:

$$r_{12} = r_{\text{AFP}_{\text{old}},\text{effort}} = 0.744502 \tag{5.5}$$

$$r_{13} = r_{\text{AFP}_{\text{new}},\text{effort}} = 0.703164 \tag{5.6}$$

$$r_{23} = r_{\text{AFP}_{\text{old}},\text{AFP}_{\text{new}}} = 0.914336 \tag{5.7}$$

$$N = 91 \tag{5.8}$$

This calculation produces a *t* value of 1.4081. At 88 degrees of freedom, the difference by which $r_{\text{AFPold,effort}}$ exceeds $r_{\text{AFPnew,effort}}$ is significant at the .10 level but not the .05 level. Further, the desired outcome was that the proposed "new" method of calculating the adjusted function point value from unadjusted function points would produce results more closely correlated with actual development effort than are results obtained using the existing method (i.e., the alternate hypothesis was that $r_{\text{AFPnew,effort}}$ would exceed $r_{\text{AFPold,effort}}$). Clearly, then, the null hypothesis

$$\text{H}_0\text{: } r_{\text{AFP}_{\text{new}},\text{effort}} = r_{\text{AFP}_{\text{old}},\text{effort}} \tag{5.9}$$

cannot be rejected.

Discussion of Results

In applying the model developed using the approach described earlier to the randomly extracted test data set, no improvement was realized in the contribution of the GSCs toward correlating adjusted function point count with development effort. This was true even with the removal of constraints which were the subject of criticism. This provides an indication that the GSCs as currently defined may be inadequate in capturing all "system" (product and platform) cost driver factors in a software development project. For those engaged in software cost estimating who were already skeptical regarding the use of function points as the basis for developing cost estimates, these results obviously would tend to increase the level of skepticism.

Based on the results of the analysis described herein with the particular data set used, there is an indication that GSCs number 9 (Complex Processing) and 13 (Multiple Sites) influence development effort and cost significantly, while GSCs number 2 (Distributed Data Processing), 7 (End-User Efficiency), and 11 (Installation Ease) have virtually no effect. If similar results were to be obtained in applying the proposed approach to other data sets, that would represent an important contribution to any initiative to "overhaul" the GSCs as currently defined, in the sense of indicating GSCs which should be retained or eliminated.

Boehm (1997, 35-37), in the COCOMO II Model, identifies a total of eight (five product and three platform) effort multipliers (cost drivers) which are based on "system" characteristics. However, as indicated earlier, the 14 function point GSCs "map" onto only three of these. This would support the contention that there are system characteristics that the GSCs do not capture which do (or can) affect development effort and cost.

Clearly, there are personnel and environmental considerations (drivers) which affect cost but which cannot be considered intrinsic characteristics of the system. DeMarco and Lister (1987) discuss such factors in detail, and Boehm (1997, 37-40) incorporates these considerations into the COCOMO II model. As a rough indicator of the relative influence afforded system characteristics (product and platform) in COCOMO II, as opposed to personnel and environmental considerations, the maximum possible contribution of each of these two groupings of factors on the estimated cost can be determined. If the eight product and platform cost drivers were assessed at their maximum values, they would result in a multiplier of 15.76 (as opposed to a multiplier of 1.00 if all were assessed at the "nominal" value); if the nine personnel and environmental cost drivers were similarly assessed at their maximum values, they would result in a multiplier of 9.48 (as opposed to 1.00 for "nominal). In other words, Boehm, in COCOMO II, while assigning greater potential weight to system (product and platform) characteristics, still assigns significant weight to personnel and environmental considerations. The function point GSCs clearly do not reflect such considerations, nor was it so intended, and this can obviously be a source of variation in effort and cost from project to project.

In examining the values of the n_i which resulted from the regression analysis and conversion from the logarithmic form described earlier, it was noted that the regression coefficients for only two of them were significant at the five percent level. It is possible that this resulted from the specific characteristics of the data set used to calculate the multipliers. It is also possible that some GSCs covaried with those GSCs whose regression coefficients were significant. Finnie, Wittig, and Desharnais (1997), reporting on research that used 299 of the 301 projects used in this research effort, in addressing claims of ambiguity as well as incompleteness levied against the GSCs as currently defined, conclude that "it appears that there is considerable covariance and that a number of the factors are difficult to separate" (43). Symons (1988, 4) similarly concludes "some of the factors, as currently defined, appear to overlap ... ; some reshuffling of the factors appears desirable." It was argued earlier that the GSCs as currently designed do not capture all of the system characteristics which have the potential to affect development effort and cost.

Conclusions

An obvious conclusion that can be drawn from the results of this effort is, for the data sample used for the analysis, the proposed multiplicative model for calculating the function point value adjustment factor did not result in an improvement over the additive model currently in use.

There are two areas upon which one should focus in order to attempt to assess the underlying reasons for the outcome realized herein. The first concerns the degree to which the data used for this analysis represent the modern software development environment in general. The second is the question of whether the general system characteristics, as currently defined, adequately capture the effects of potential system (i.e., product and platform) cost drivers.

First addressing the adequacy of the data used for this research to represent software development in general: it has been the experience of this researcher with an earlier research attempt as well as with this effort that obtaining software development project data, especially data relating to development effort, cost, and productivity, is quite difficult. There is an extreme reluctance in the industry to share such information. It was thus necessary to work with those data which were available. The data available to this researcher were, with the exception of two projects, obtained from a single source, Jean-Marc Desharnais of the University of Quebec at Montreal. The data represent software development projects which were completed over the period 1982 through 1990. Eighty-one of the projects were used by Desharnais (1988) as the source of data for his Master's Thesis, and all 299 projects were used by Finnie, Wittig, and Desharnais (1997), for the data used in their research. Data from the 299 projects were obtained from only 17 organizations, and all 299 development projects were for MIS (business) applications (the two projects for which data were provided by Boland were for scientific/engineering applications (Boland 1998)). Desharnais himself accomplished the function point counts for 168 of the 299 projects, although this should not present a problem, based on the research reported by Kemerer (1993) pertaining to interrater reliability in the counting of function points.

While 301 projects should be a sufficient number upon which to base meaningful conclusions, clearly 299 of them cannot be considered to be representative of a cross-section of modern software development projects, in view of the time period during which they were completed, the limited number of organizations from which they were drawn, and the fact that they all lie in the management information system (MIS) application domain. Therefore, no conclusion can be reached as to whether the removal of constraints on determining the value of the value adjustment factor, using the proposed new approach, would have resulted in an improvement (i.e., better correlation of the adjusted function point count with development effort) over the current method for non-MIS applications.

As noted earlier, obtaining data for software development projects is a challenge. However, if adequate data can be collected, the proposed alternative (multiplicative) method should be re-evaluated using data from much more recently completed projects, representing a cross-section of application domains as well as a greater number of software development organizations. Only then can the results be generalized to the overall software development community.

With respect to the adequacy of the 14 general system characteristics, as currently defined, in capturing the effects of potential system (i.e., product and platform) cost drivers, there are a number of factors to consider: first, the proposed modification to the treatment of the 14 general system characteristics would have the effect of easing, not tightening or adding, constraints on the contribution of a given GSC to determining the value adjustment factor. Intuitively, then, the fact that no improvement in correlation of the adjusted function point count with development effort and cost was realized would seem to be in spite of, and not because of, the proposed multiplicative approach. This would tend to point to inadequacies with the GSCs themselves, rather than the proposed new approach.

Another consideration is the amount of time which has elapsed since these GSCs were last revised. Ten system characteristics were introduced with the original advent of function points in 1979. The current 14 were established in 1984 and have been used unchanged ever since. However, the software development environment has undergone dramatic changes since that period, perhaps indicating a need to review and revise the current GSCs.

As noted earlier, the 14 function point GSCs, when mapped onto the eight of Boehm's 17 COCOMO II cost drivers which can reasonably be considered system characteristics, correlate to only three of the COCOMO II cost drivers. This suggests (1) that there are likely effort and cost driver factors which are not captured by the 14 existing GSCs, and (2) that the function point GSCs may not all be substantially independent of one another. There may as well be significant interaction effects among cost drivers. As previously cited, for the 299 software development projects discussed above, which served as the data source for Finnie, Wittig, and Desharnais (1997) and in large measure for the research described herein, Finnie, Wittig, and Desharnais found that "it appears that there is considerable covariance and that a number of the factors [the 14 GSCs] are difficult to separate." These factors further underscore the need to review and revise the existing GSCs, notwithstanding the fact that doing so will likely render unusable (for cost estimating purposes) some of the data which have been accumulated for past development projects.

From discussions held with various experts concerned with software size measurement and software development cost estimating while defining and carrying out the research described herein, it is the sense of this researcher that two obstacles to improving the use of function points as the basis for cost estimation and prediction are: (1) the extreme reluctance of individuals and organizations engaged in software development to provide data for research, even with anonymity assured; and (2) the lack of dialogue between the proponents of the function points approach and the developers of COCOMO II. It is the opinion of this researcher that IFPUG (and perhaps similar organizations) could serve as catalysts toward a freer flow of data to support research while ensuring the security of business sensitive information. A teaming of IFPUG with the academic community may help; IFPUG has established an Academic Affairs Committee and has begun making overtures to academic institutions where there may be an interest in pursuing research in this area. It is also the opinion of this researcher that Boehm's team at the University of Southern California should either modify COCOMO II to accommodate function points directly as a size metric or develop a separate function points variant of COCOMO II, rather than requiring the use of an industry-average "conversion factor" to equate function points to lines-of-code prior to applying the cost estimating model.

Limitations of the Research

The most serious limitation of the research is the unavailability of data which represent a cross-section of modern software development efforts. As indicated earlier, of the 301 projects from which data were used in this effort, 299 were provided by the same source. The 299 development projects were completed within a total of only 17 organizations over the period 1982 through 1990. All were essentially management information system (business) applications (Desharnais 1998). The remaining two projects were scientific/engineering applications from a single organization and were completed in 1997 (Boland 1998).

It has been the experience of this researcher that software development project data of the type needed to complete research of this nature (i.e., software development effort and cost) are extremely difficult to obtain. Gordon Lundquist, the Director of Applied Programs for the International Function Point Users Group (IFPUG), advised this researcher that, even though he represented a sanctioned, recognized industry group, he had a great deal of difficulty obtaining such data, even with guarantees of non-attribution, from IFPUG member organizations. The concern within the industry, which is quite competitive, is one of providing even the slightest insight by a competitor into one's costs or productivity. The problem is even worse for individuals seeking to obtain data for research. Also as indicated above, perhaps a stronger alliance and better cooperation among IFPUG, its member organizations, and the academic community are needed.

Avenues for Further Research

The results obtained herein indicate a number of areas where further research may be appropriate and useful.

First, provided the necessary data can be collected so that a cross-section of modern software development projects is represented, the approach proposed above should be applied to such data. This will serve either to demonstrate that the proposed approach has merit or (as is more likely the case) to reinforce the conclusion that the function point general system characteristics do not incorporate all of the system characteristics which drive effort and costs in modern software development projects.

Whether or not similar results are experienced in such a reassessment of the proposed approach, the function point general system characteristics, as currently defined, should be carefully re-evaluated and ultimately redefined as appropriate to ensure (1) that they are substantially independent of one another and (2) that they adequately capture the effects of all system (i.e., product and platform) characteristics which drive effort and therefore costs.

The research effort described herein accepts the procedure for determining the raw, or unadjusted, function point count as a "given," in order to focus on the effects of the general system characteristics upon the final (adjusted) function point count as it relates to development cost. In actuality, the techniques used to determine the raw function point count can, and should, be subjected to the same type of scrutiny. After all, Albrecht, by his own admission, arrived at his method for calculating function points by "debate and trial" (Albrecht 1979, 85). Abran and Robillard (1994, 172) point out that function points are an algorithmic metric, and therefore have the problems inherent in any algorithmic (or synthetic) metrics system. They observe that algorithmic metrics are difficult to interpret and state that "the reasons for the assignments of specific values (weights) are not clear." Additionally, Finnie, Wittig, and Desharnais (1997, 43) advise that "the weighting scheme used in Function Point Analysis [for calculating unadjusted function points] is in need of some reassessment."

Clearly there are factors beyond system characteristics which affect effort and costs, specifically personnel and environmental considerations. While system characteristics obviously pertain to the specific system under development, personnel and environmental factors are characteristics of the software development organization. A proposed avenue of research is to seek to develop guidelines for use by individual organizations in assessing their "organizational characteristics" which would attempt to capture personnel and environmental considerations which affect development effort, in order to complement the assessment of the effects of system (product and platform) characteristics. To some extent, the need for incorporating such organizational considerations into one's cost estimating approach is reflected by the emphasis on the part of several authors (Kemerer 1987, 427; Matson, Barrett, and Mellichamp 1994, 284; Garmus and Herron 1996a, 141, and 1996b, 65; Jones 1996a, 3; Gaffney 1996, 8) on the importance for an organization to "calibrate" its function points-based cost estimates to its own historical data. However, since conditions may vary from project to project within an organization, an approach for assessing the organizational characteristics pertaining to a given software development project, such as is suggested here, would provide a tool for use by organizations in accommodating such variations. In view of the nature of the COCOMO II Personnel and Environmental cost drivers, it is likely that significant work in this direction has already been accomplished by Boehm and his team at the University of Southern California.

Even though the function points approach in general and the method of weighting the counts of the five basic entities have been the objects of criticism, the technique as currently applied still enjoys widespread acceptance. What may be needed is for COCOMO II or a variant thereof to be revamped so that unadjusted function points are used as the principal software size metric, rather than retaining the required lines-of-code conversion. In addition, the burden should fall upon IFPUG to explore alternatives to the current weighting scheme used in calculating unadjusted function points and to approve and promulgate changes to existing counting practices thus identified when it makes sense to do so. Closer dialogue is needed between Boehm's team and IFPUG than has existed to date. Particular consideration should be given to the possible advantages of replacing the current additive model for calculating the value adjustment factor with a multiplicative one, similar to that described herein.

REFERENCE LIST

Abran, Alain, and Pierre N. Robillard. 1994. Function points: A study of their measurement processes and scale transformations. Journal of Systems Software 25: 171-184.

Albrecht, Allan J. 1979. Measuring application development productivity. Proceedings of the IBM Applications Development Symposium, GUIDE/SHARE, Monterey, CA, October 14-17: 83-92.

Albrecht, Allan J., and John E. Gaffney, Jr. 1983. Software function, source lines of code, and development effort prediction: A software science validation. IEEE Transactions on Software Engineering SE-9 (November): 639-647.

Bernstein, Lawrence, and Alex Lubashevsky. 1995. Living with function points. CrossTalk: The Journal of Defense Software Engineering 8 (November/December): 14-18.

Boehm, Barry W. 1981. Software engineering economics. Englewood Cliffs, NJ: Prentice-Hall.

________. 1997. COCOMO II model definition manual, version 1.4. Los Angeles: University of Southern California.

Boehm, Barry W., Bradford Clark, Ellis Horowitz, Chris Westland, Ray Madachy, and Richard Selby. 1995. Cost models for future software life cycle processes: COCOMO 2.0. In Annals of software engineering special volume on software process and product measurement, ed. J.D. Arthur and S.M. Henry. Amsterdam: J.C. Baltzer AG.

Boland, Dillard. 1998. Computer Sciences Corporation, Lanham, MD. Telephone conversation with author. April 22.

Conte, S.D., H.E. Dunsmore, and V.Y. Shen. 1986. Software engineering metrics and models. Menlo Park, CA: Benjamin/Cummings.

DeMarco, Tom, and Timothy R. Lister. 1987. Peopleware: productive projects and teams. New York: Dorset House.

Desharnais, Jean-Marc. 1988. Statistical analysis on the productivity of data processing with development projects using the function point technique. Masters thesis. University of Quebec at Montreal.

________. 1998. University of Quebec at Montreal. Function point data. Electronic mail dated April 21.

Downie, N.M., and R.W. Heath. 1974. Basic statistical methods. New York: Harper and Row.

Dreger, J. Brian. 1989. Function point analysis. Englewood Cliffs, NJ: Prentice Hall.

Finnie, G.R., G.E. Wittig, and Jean-Marc Desharnais. 1997. Reassessing function points. Australian Journal of Information Systems. 4 (May): 39-45.

Gaffney, John E., Jr. 1996. Software cost estimation using simplified function points. Proceedings of the Spring Software Technology Conference. Ogden, UT: United States Air Force Software Technology Support Center.

Garmus, David, and David Herron. 1996a. Measuring the software process: A practical guide to functional measurements. Upper Saddle River, NJ: Prentice Hall.

________. 1996b. Effective early estimation. Software Development (July): 57-65.

Giles, Alan E., and Dennis Barney. 1995. Metrics tools: Software cost estimation. CrossTalk: The Journal of Defense Software Engineering 8 (June): 7-10.

Jamieson, Paula, immediate past president, International Function Point Users Group (IFPUG). 1997. Telephone conversation with author. September 29.

Jones, Capers. 1991. Applied software measurement: Assuring productivity and quality. New York: McGraw-Hill.

________. 1996a. Applied software measurement: Assuring productivity and quality (second edition). New York: McGraw-Hill.

_______. 1996b. Estimating and measuring object-oriented software (draft). Prepared for publication in American Programmer. Burlington, MA: Software Productivity Research, Inc.

Kemerer, Chris F. 1987. An empirical validation of software cost estimation models. Communications of the ACM 30 (May): 416-429.

_______. 1993. Reliability of function points measurement: A field experiment. Communications of the ACM 36 (February): 85-97.

Kemerer, Chris F., and Benjamin S. Porter. 1992. Improving the reliability of function point measurement: An empirical study. IEEE Transactions on Software Engineering 18 (November): 1011-1024.

Keyes, Jessica. 1992. New metrics needed for new generation. Software Magazine (May): 42-56.

Low, Graham C., and D. Ross Jeffery. 1990. Function points in the estimation and evaluation of the software process. IEEE Transactions on Software Engineering 16 (January): 64-71.

Major, Melissa L. 1996. A qualitative analysis of two requirements capturing techniques for estimating the size of object-oriented software projects. Masters thesis. Clemson University.

Matson, Jack E., Bruce E. Barrett, and Joseph M. Mellichamp. 1994. Software development cost estimation using function points. IEEE Transactions on Software Engineering 20 (April): 275-287.

Mukhopadhyay, Tridas, and Sunder Kekre. 1992. Software effort models for early estimation of process control applications. IEEE Transactions on Software Engineering 18 (October): 915-924.

Navlakha, Jainendra K. 1990. Choosing a software cost estimation model for your organization: A case study. Information and Management 18: 255-261.

Oskarsson, Osten, and Robert L. Glass. 1996. An ISO 9000 approach to building quality software. Upper Saddle River, NJ: Prentice Hall.

Rehesaar, Hugo. 1997. ISESS '97 FSM workshop. Electronic mail dated February 4.

Srinivasan, Bala, and Geoff Martin. 1994. MONSET - A prototype software development estimating tool. In Proceedings of the Third Symposium on Assessment of Quality Software Development Tools in Washington, DC, June 7-9, 1994, by the IEEE Computer Society Technical Committee on Software Engineering. Los Alamitos, CA: IEEE Computer Society Press, 70-82.

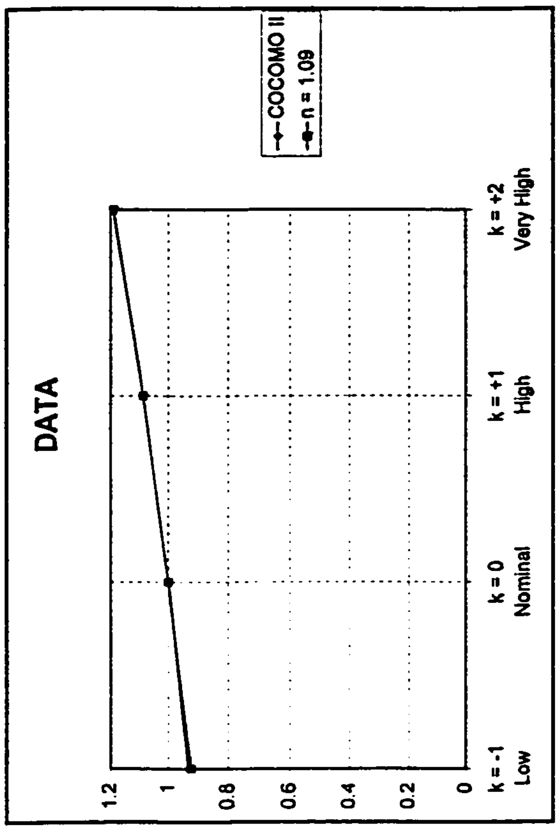
Symons, Charles R. 1988. Function point analysis: Difficulties and improvements. IEEE Transactions on Software Engineering 14 (January): 2-11.

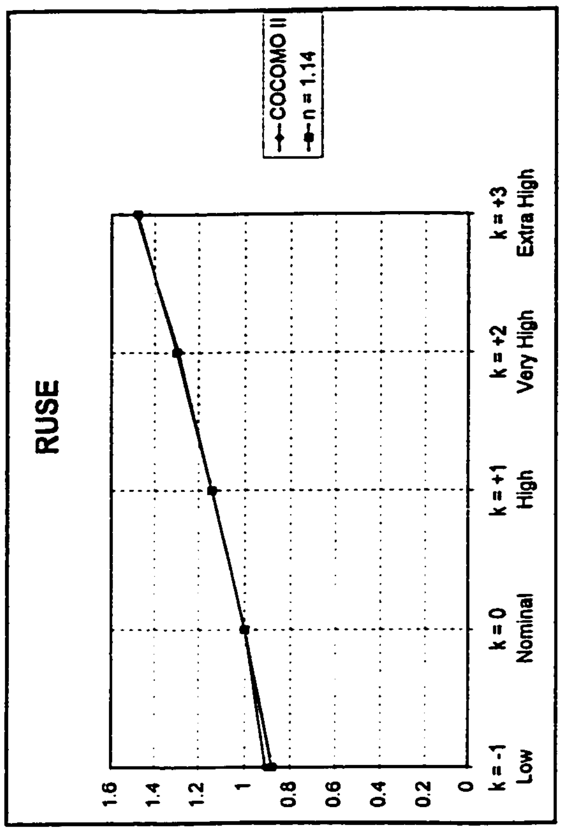
Wellman, Frank. 1992. Software costing. New York: Prentice Hall.

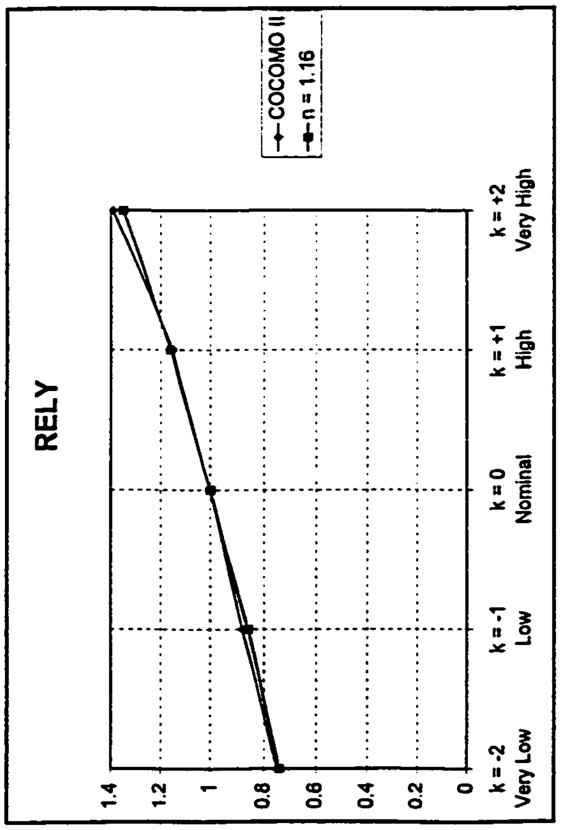
Whitmire, Scott A. 1995. An introduction to 3D function points. Software Development (April): 43-53.

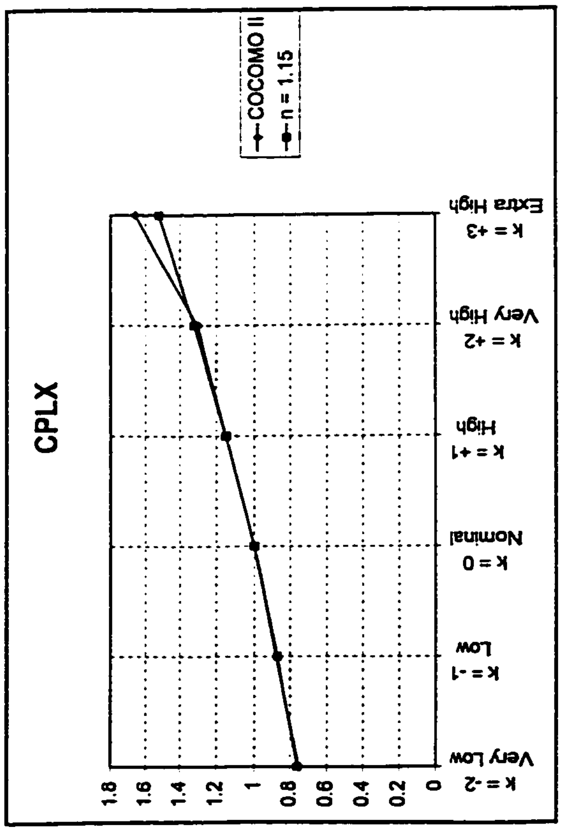
APPENDIX A

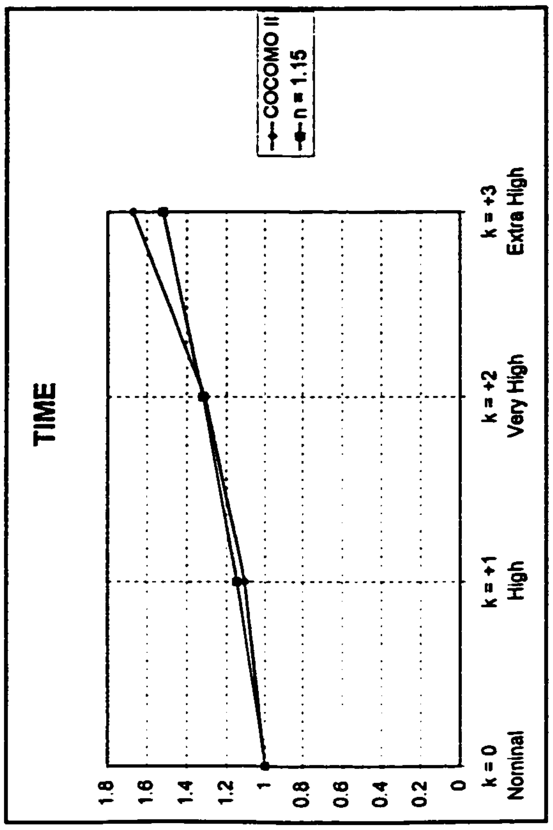
COMPARISON OF COCOMO II COST DRIVER PROVISIONAL VALUES WITH CORRESPONDING VALUES OBTAINED USING PROPOSED EXPONENTIAL FORM

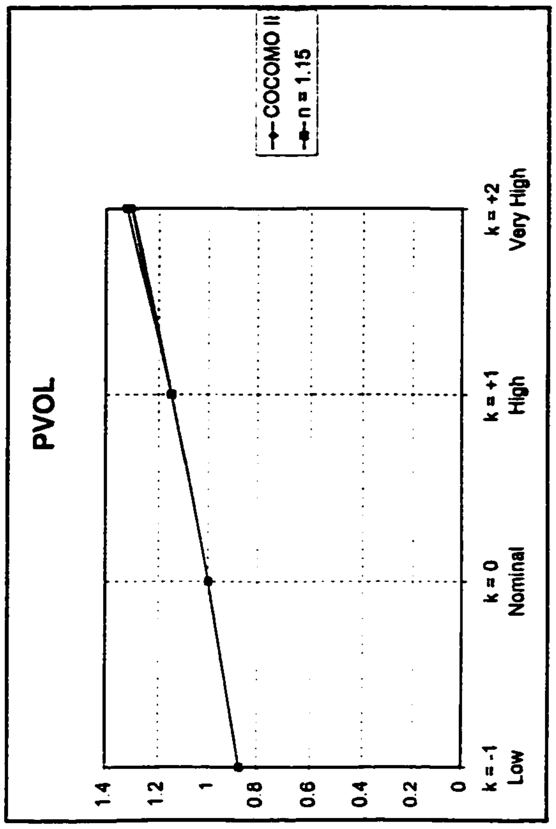

DATA
COCOMO II
n = 1.09
1.2
1
0.8
0.6
0.4
0.2
0
k = -1 Low
k = 0 Nominal
k = +1 High
k = +2 Very High

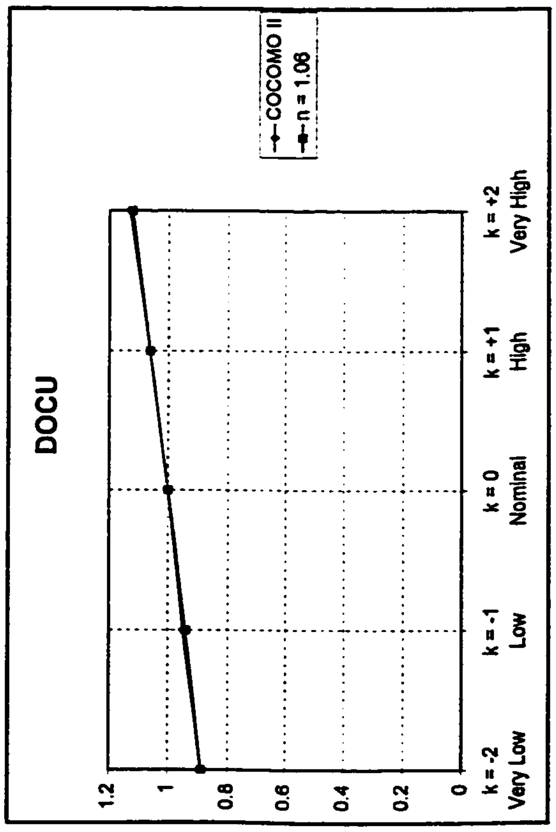

RUSE
COCOMO II
n = 1.14
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
k = -1 Low
k = 0 Nominal
k = +1 High
k = +2 Very High
k = +3 Extra High

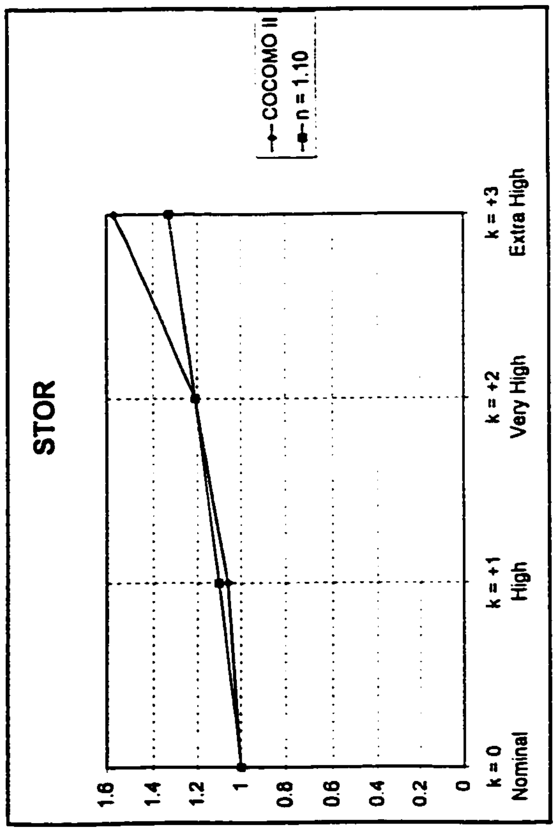

RELY
COCOMO II
n = 1.16
1.4
1.2
1
0.8
0.6
0.4
0.2
0
k = -2 Very Low
k = -1 Low
k = 0 Nominal
k = +1 High
k = +2 Very High

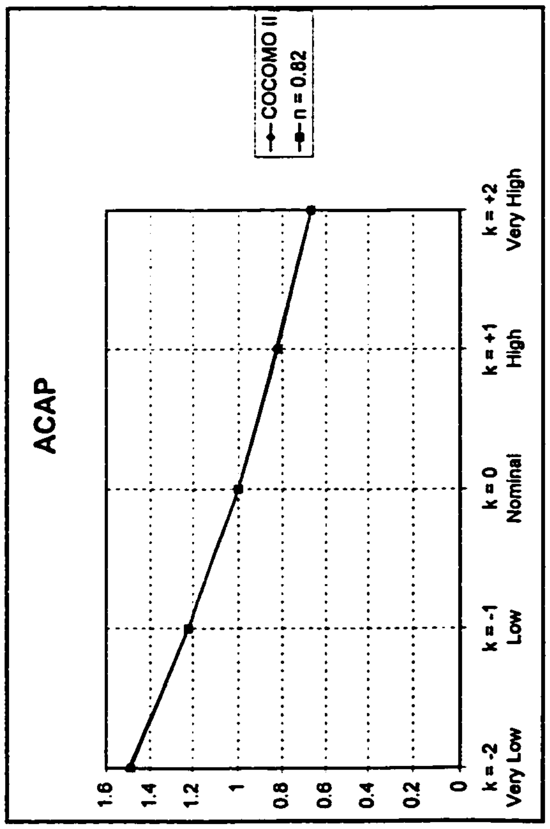

CPLX
COCOMO II
n = 1.15
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
k = -2 Very Low
k = -1 Low
k = 0 Nominal
k = +1 High
k = +2 Very High
k = +3 Extra High

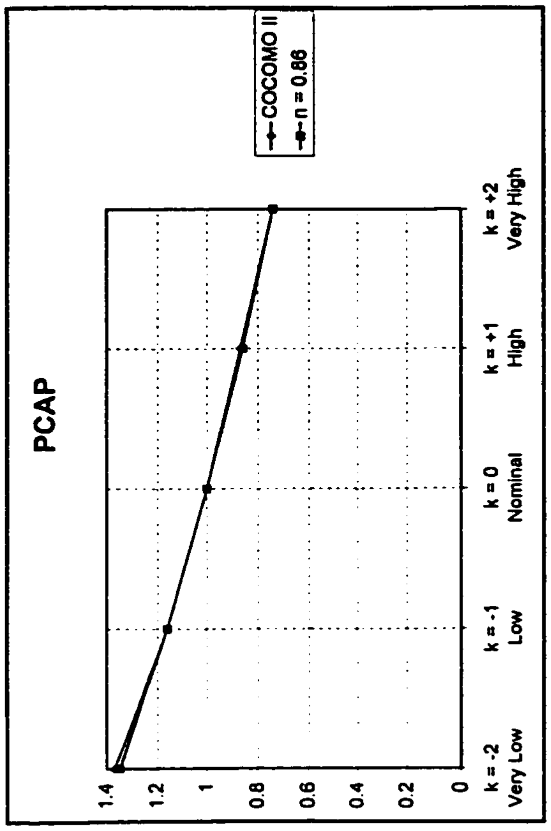

TIME
COCOMO II
n = 1.15
k = 0 Nominal
k = +1 High
k = +2 Very High
k = +3 Extra High
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8

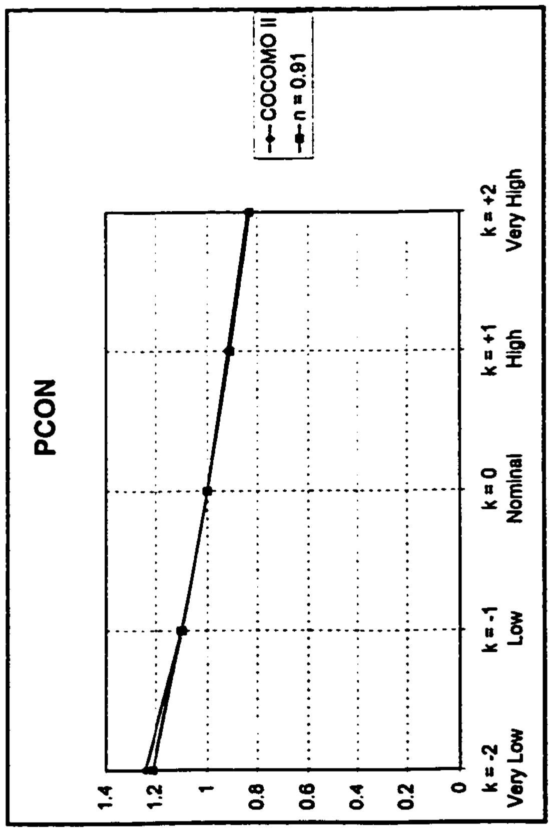

PVOL
COCOMO II
n = 1.15
k = -1 Low
k = 0 Nominal
k = +1 High
k = +2 Very High
0
0.2
0.4
0.6
0.8
1
1.2
1.4

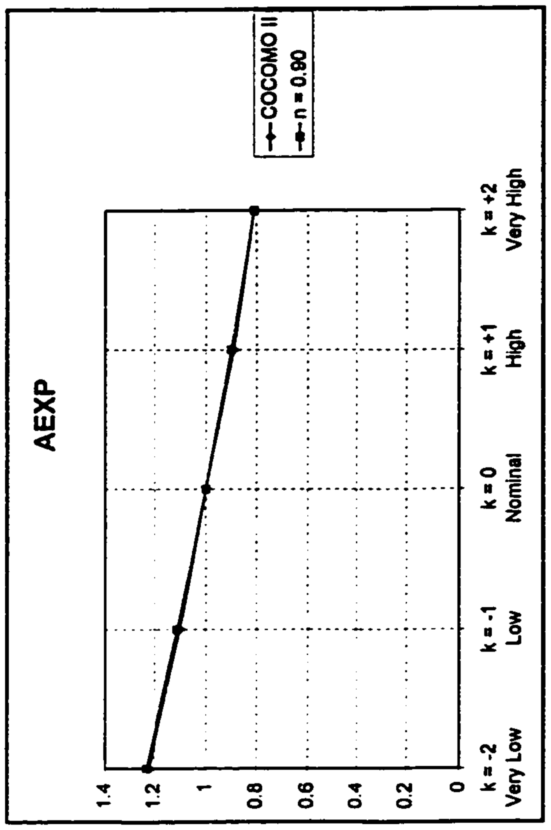

DOCU
COCOMO II
n = 1.06
k = -2 Very Low
k = -1 Low
k = 0 Nominal
k = +1 High
k = +2 Very High
0
0.2
0.4
0.6
0.8
1
1.2

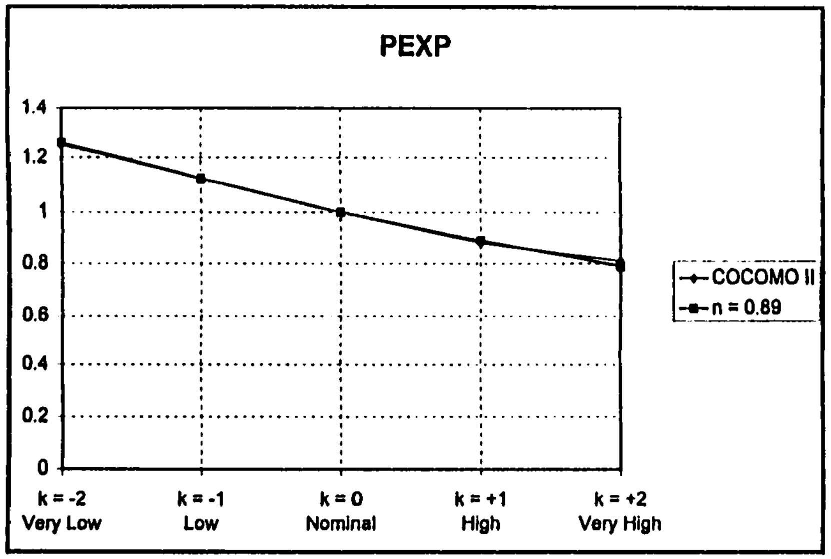

STOR
COCOMO II
n = 1.10
k = 0 Nominal
k = +1 High
k = +2 Very High
k = +3 Extra High
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6

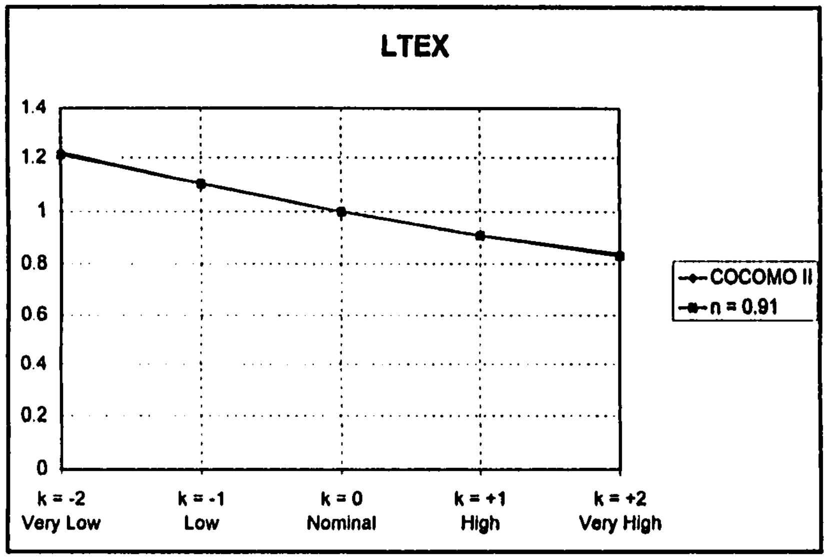

ACAP
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
k = -2
Very Low
k = -1
Low
k = 0
Nominal
k = +1
High
k = +2
Very High
COCOMO II
n = 0.82

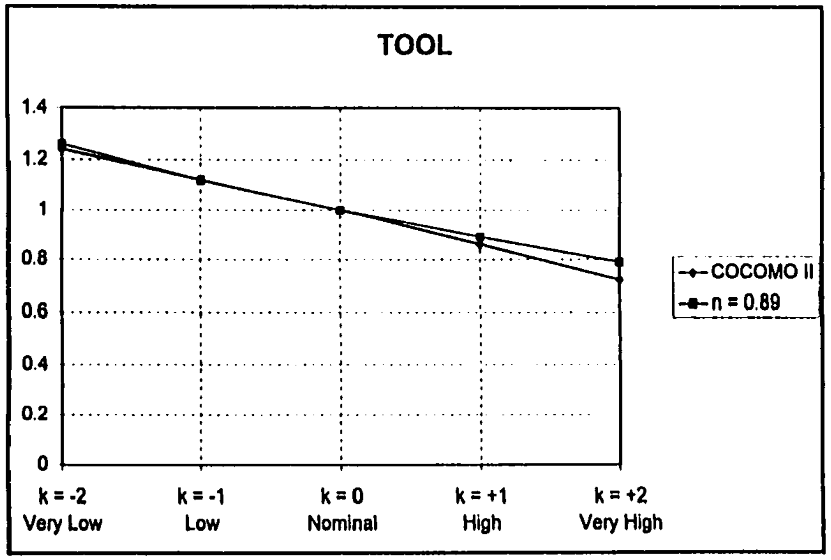

PCAP
1.4
1.2
1
0.8
0.6
0.4
0.2
0
k = -2
Very Low
k = -1
Low
k = 0
Nominal
k = +1
High
k = +2
Very High
COCOMO II
n = 0.86

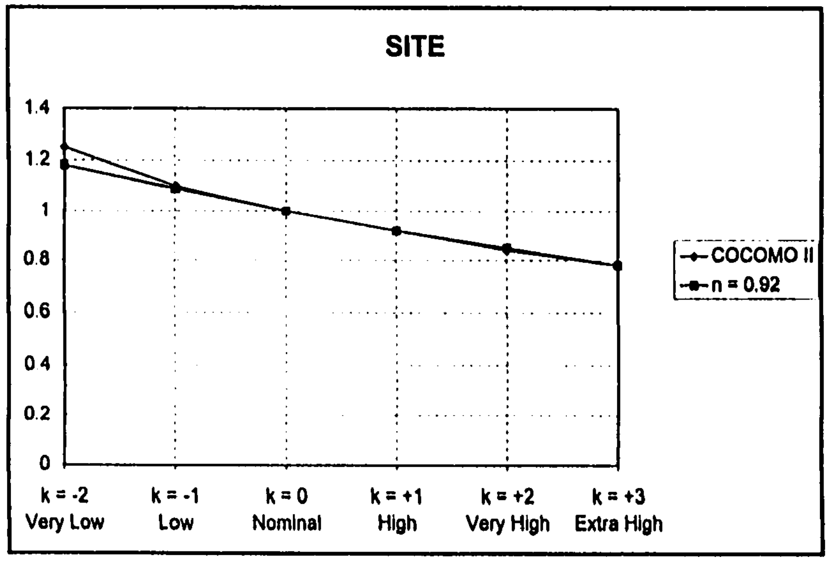

PCON
1.4
1.2
1
0.8
0.6
0.4
0.2
0
k = -2
Very Low
k = -1
Low
k = 0
Nominal
k = +1
High
k = +2
Very High
COCOMO II
n = 0.91

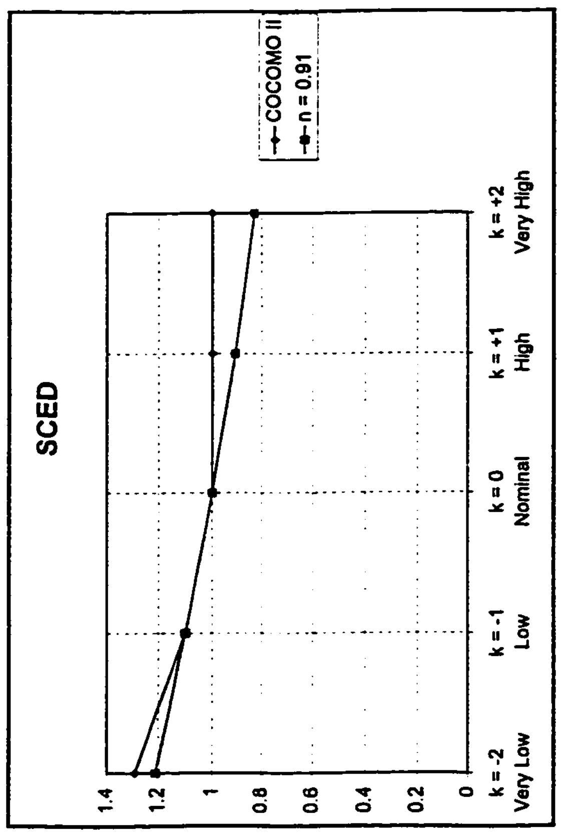

AEXP
1.4
1.2
1
0.8
0.6
0.4
0.2
0
k = -2
Very Low
k = -1
Low
k = 0
Nominal
k = +1
High
k = +2
Very High
COCOMO II
n = 0.90


PEXP
1.4
1.2
1
0.8
0.6
0.4
0.2
0
k = -2 Very Low
k = -1 Low
k = 0 Nominal
k = +1 High
k = +2 Very High
COCOMO II
n = 0.89


LTEX
1.4
1.2
1
0.8
0.6
0.4
0.2
0
k = -2 Very Low
k = -1 Low
k = 0 Nominal
k = +1 High
k = +2 Very High
COCOMO II
n = 0.91


TOOL
1.4
1.2
1
0.8
0.6
0.4
0.2
0
k = -2 Very Low
k = -1 Low
k = 0 Nominal
k = +1 High
k = +2 Very High
COCOMO II
n = 0.89


SITE
1.4
1.2
1
0.8
0.6
0.4
0.2
0
k = -2 Very Low
k = -1 Low
k = 0 Nominal
k = +1 High
k = +2 Very High
k = +3 Extra High
COCOMO II
n = 0.92


SCED
COCOMO II
n = 0.91
1.4
1.2
1
0.8
0.6
0.4
0.2
0
k = -2 Very Low
k = -1 Low
k = 0 Nominal
k = +1 High
k = +2 Very High

APPENDIX B

PROJECT DATA FOR 301

SOFTWARE DEVELOPMENT PROJECTS

Project No	Effort (mh)	UFP	GSC #1	GSC #2	GSC #3	GSC #4	GSC #5	GSC #6	GSC #7	GSC #8	GSC #9	GSC #10	GSC #11	GSC #12	GSC #13	GSC #14	AFP_{old}
1	5152	204	5	0	3	3	2	5	4	3	0	4	3	1	0	1	202
2	2569	168	5	0	2	3	1	5	2	3	0	0	3	1	0	0	151
3	805	107	5	0	2	3	0	0	5	0	0	2	0	1	0	0	89
4	3913	227	4	0	2	3	0	5	1	4	4	0	1	1	0	0	204
5	3829	296	5	0	1	1	1	5	5	3	3	2	4	0	0	0	281
6	5635	328	5	0	2	1	1	5	4	4	2	4	1	0	0	4	321
7	2149	241	5	1	1	1	1	1	4	2	3	3	0	0	0	2	214
8	2821	193	4	0	5	1	5	5	5	3	2	4	2	0	0	4	203
9	3040	179	5	0	1	0	3	5	3	3	3	1	1	1	0	3	168
10	6895	271	3	0	3	0	1	5	1	5	3	3	5	1	0	0	257
11	4908	320	0	0	1	0	1	1	1	0	3	1	3	3	0	0	253
12	7721	254	5	0	1	0	1	5	5	3	5	1	5	0	0	5	257
13	18314	375	5	0	5	5	3	5	3	3	5	0	5	3	0	5	420
14	1197	101	5	0	5	0	5	5	3	5	5	1	1	1	0	0	102
15	4273	99	0	0	5	0	3	0	3	0	5	1	3	1	0	0	85
16	5710	447	3	0	3	0	1	5	3	3	3	3	3	3	0	0	425
17	5470	384	0	0	5	0	3	0	3	0	5	1	3	1	0	0	313
18	2155	75	5	0	5	0	5	5	3	5	5	1	1	1	0	0	76
19	6368	158	0	0	3	0	3	0	1	0	3	1	3	1	0	3	131
20	10210	163	0	0	3	0	1	1	1	0	3	1	3	3	0	0	132
21	5985	269	3	0	3	0	3	5	1	3	1	1	1	3	0	3	247
22	1784	175	5	0	1	0	3	5	3	3	3	1	1	1	0	3	164
23	4620	248	0	0	3	0	3	0	3	0	3	1	3	1	0	3	211
24	2514	244	1	0	5	0	5	3	1	0	5	1	3	3	0	0	224
25	501	186	3	0	0	0	0	5	0	0	0	0	0	0	0	0	136
26	1259	164	4	0	0	0	0	2	0	0	0	1	0	0	0	0	118
27	11493	449	3	0	1	0	2	4	1	2	3	3	1	3	3	3	422
28	1636	127	0	0	0	0	0	0	0	0	0	0	0	0	0	0	83
29	14700	348	5	0	4	3	5	5	2	3	0	4	3	0	0	0	345
30	5700	339	2	0	5	0	0	0	2	0	2	1	1	4	0	1	281
31	1050	337	5	0	5	0	4	5	4	5	0	3	4	3	1	0	350
32	3000	560	5	0	0	0	0	5	0	0	0	0	1	0	0	0	426
33	260	75	0	0	2	2	0	0	0	0	0	2	2	0	0	1	56
34	6557	367	5	0	2	0	0	5	3	0	1	2	3	0	0	1	319
35	5865	560	5	0	3	0	0	5	4	0	3	0	0	0	1	0	482
36	1830	264	5	0	0	0	0	5	0	0	0	0	4	0	0	0	209
37	12630	1003	5	0	5	0	5	5	0	4	4	3	3	0	0	3	1023
38	820	201	1	0	1	0	0	1	0	0	0	0	1	0	0	0	139
39	4830	220	5	0	4	2	0	1	4	0	5	1	2	2	0	2	205
40	5568	328	5	0	0	2	2	5	2	0	1	0	1	1	0	1	279
41	6129	397	5	0	1	1	1	5	0	0	1	0	0	0	1	0	318
42	247	172	5	1	0	0	0	5	0	0	1	0	3	0	0	0	138
43	4291	681	5	0	0	0	5	5	3	5	0	2	4	0	3	0	661

Project No	Effort (mh)	UFP	GSC #1	GSC #2	GSC #3	GSC #4	GSC #5	GSC #6	GSC #7	GSC #8	GSC #9	GSC #10	GSC #11	GSC #12	GSC #13	GSC #14	AFP_{old}
44	19724	569	0	0	0	0	0	2	0	2	4	0	0	2	0	5	467
45	60142	851	3	4	4	5	4	5	5	5	5	3	0	0	0	5	996
46	5670	355	2	0	0	0	2	5	3	4	2	3	0	2	0	2	320
47	5535	256	4	0	3	0	0	2	4	0	3	0	0	2	0	5	225
48	4874	228	0	0	3	0	0	1	1	0	3	0	3	0	0	0	173
49	4604	197	0	0	0	0	0	0	0	0	2	0	0	0	0	5	142
50	1944	82	0	0	0	0	0	0	0	0	2	0	0	0	0	0	55
51	1526	84	0	0	0	0	0	0	0	0	2	0	0	2	0	0	58
52	3780	108	0	0	0	0	0	0	0	0	2	2	0	2	0	1	78
53	4320	147	0	0	2	0	0	0	0	0	3	1	0	2	0	5	115
54	5049	183	0	0	0	0	0	0	0	0	0	0	0	0	0	0	119
55	2362	140	0	0	0	0	0	0	0	0	0	0	0	0	0	0	91
56	4792	139	0	0	0	0	0	0	0	0	2	3	3	0	0	0	101
57	1256	108	0	0	0	0	2	0	0	0	2	0	0	0	0	0	80
58	12164	208	0	0	3	0	2	3	3	2	3	2	1	0	0	1	187
59	4658	261	0	0	1	0	0	0	0	0	2	0	0	2	0	2	188
60	5926	196	0	0	0	0	0	0	0	0	2	2	1	0	0	3	143
61	4442	142	0	0	2	0	0	0	0	0	2	5	0	0	0	5	112
62	4792	149	0	0	0	0	0	0	0	0	2	0	2	1	0	1	106
63	7654	431	0	0	0	0	0	1	0	0	3	0	0	0	0	1	302
64	41445	669	3	0	4	1	3	3	4	1	5	0	3	3	1	3	696
65	2160	149	4	3	2	3	0	1	4	0	1	0	0	0	0	3	136
66	9801	204	4	4	0	0	0	0	0	0	3	0	1	2	0	0	161
67	3551	120	0	0	3	0	0	0	0	0	3	0	3	0	0	5	95
68	8235	167	0	0	1	0	1	0	0	0	5	0	0	1	0	0	122
69	4050	155	0	0	0	0	0	0	0	0	2	0	1	2	0	0	108
70	67594	1039	5	0	5	0	5	1	5	3	5	0	4	3	0	5	1101
71	10759	328	2	0	5	0	0	0	0	0	5	0	0	4	0	0	272
72	7614	460	0	0	0	0	0	0	0	0	0	0	0	0	0	0	299
73	18644	632	4	3	3	0	0	0	0	0	2	2	1	1	0	2	525
74	15916	1003	5	2	1	0	0	0	0	0	4	1	3	2	1	1	853
75	5386	477	4	3	3	0	0	0	0	0	2	2	1	1	0	2	396
76	11367	459	2	0	2	0	0	5	5	3	4	0	0	2	0	5	450
77	6359	220	0	0	2	0	0	0	0	0	5	0	0	2	0	1	172
78	1904	126	0	0	0	0	1	0	0	0	5	0	2	2	0	3	98
79	8883	237	3	0	0	0	0	0	0	0	3	0	0	2	0	2	178
80	25609	1003	0	0	2	0	2	3	0	2	2	0	0	0	0	2	802
81	12339	599	0	0	1	0	0	1	0	0	2	0	2	0	0	1	431
82	7155	179	0	0	3	0	0	2	3	2	4	3	2	2	2	3	163
83	10962	352	0	0	3	0	0	1	0	0	4	0	3	0	0	0	268
84	7169	419	0	0	2	0	0	0	0	0	2	0	0	0	0	0	289
85	6251	264	1	0	0	0	0	0	0	0	3	0	0	0	0	0	182
86	8249	306	4	0	0	0	0	3	2	1	2	0	1	3	0	1	263

Project No	Effort (mh)	UFP	GSC #1	GSC #2	GSC #3	GSC #4	GSC #5	GSC #6	GSC #7	GSC #8	GSC #9	GSC #10	GSC #11	GSC #12	GSC #13	GSC #14	AFP_{old}
87	7601	347	0	0	0	0	0	0	0	0	5	0	0	0	0	5	260
88	7074	239	0	0	5	0	0	0	0	0	5	0	0	2	0	3	191
89	4536	174	3	1	0	0	1	1	3	1	0	0	1	0	2	2	144
90	5049	477	3	0	2	0	3	5	5	4	0	0	0	2	1	2	439
91	9315	655	3	0	4	3	4	4	4	4	4	0	3	0	0	0	642
92	13703	134	5	5	5	3	1	1	4	0	5	5	3	0	0	5	150
93	6494	292	3	0	5	0	4	5	3	4	5	2	4	3	0	2	321
94	34317	798	3	0	5	0	5	5	5	4	4	0	1	3	0	3	862
95	6926	331	0	0	0	0	0	1	0	0	1	0	0	1	3	0	235
96	54716	820	4	0	5	0	4	5	4	4	5	5	5	2	0	4	959
97	3942	129	0	0	0	0	0	0	0	0	2	0	0	2	2	1	95
98	4293	341	3	3	5	1	0	5	4	4	2	0	0	2	3	4	344
99	7519	676	1	0	0	3	3	5	5	4	0	0	0	0	0	3	635
100	2565	153	0	0	0	0	0	0	0	0	2	0	0	2	0	0	113
101	7344	297	2	0	3	0	0	0	3	0	4	1	3	3	0	3	267
102	16996	425	0	0	3	0	0	0	0	0	4	2	4	3	0	3	378
103	12271	363	0	0	4	0	0	0	0	0	5	3	2	2	0	0	312
104	7884	326	3	0	5	0	4	5	3	4	5	2	4	3	0	2	359
105	4509	168	0	0	0	0	0	0	0	0	2	0	1	0	0	0	114
106	2254	78	0	0	0	0	0	0	0	0	2	0	0	0	0	0	52
107	15309	590	5	4	5	5	5	5	5	4	3	0	5	2	3	5	732
108	4860	277	5	0	5	5	5	3	3	4	3	5	5	0	0	5	313
109	1566	72	0	0	0	0	0	0	0	0	4	0	0	0	0	3	53
110	26474	820	5	0	5	2	5	5	4	4	3	0	5	3	0	5	951
111	1512	85	1	0	0	0	0	0	0	0	0	0	0	0	0	0	56
112	1053	60	2	1	0	0	1	1	1	1	0	0	0	0	1	0	44
113	3132	134	4	4	2	3	0	0	0	0	0	0	3	2	4	3	122
114	5468	134	0	0	5	0	0	0	0	0	2	0	3	5	0	0	107
115	8006	220	0	0	0	0	0	0	0	0	4	0	0	2	0	0	161
116	6656	479	0	0	0	0	0	0	0	0	3	0	0	2	0	2	345
117	5805	329	0	0	0	0	0	0	0	0	2	0	0	2	0	0	234
118	3874	173	0	0	0	0	0	0	0	0	2	0	0	2	0	0	119
119	11003	961	4	2	2	0	1	5	2	2	4	0	4	1	0	3	961
120	3992	280	5	0	1	0	0	5	1	0	0	0	0	0	0	1	218
121	20205	568	5	0	3	0	2	5	4	0	0	2	2	0	0	4	523
122	493	100	5	0	0	0	0	5	0	0	1	1	1	0	0	0	78
123	2150	176	5	0	0	0	0	5	0	0	1	1	1	0	0	0	137
124	3452	188	5	0	0	0	0	5	0	0	1	1	1	0	0	0	147
125	5522	270	5	2	0	0	0	5	0	0	0	0	0	0	0	4	219
126	2430	177	5	2	0	0	0	0	0	0	0	0	0	0	4	0	135
127	1617	162	3	0	2	3	3	5	2	2	0	2	1	3	0	0	147
128	2422	123	3	0	2	3	3	5	1	3	0	0	1	3	0	0	109
129	2282	112	0	0	2	3	2	0	1	0	5	0	1	3	0	2	94

Project No	Effort (mh)	UFP	GSC #1	GSC #2	GSC #3	GSC #4	GSC #5	GSC #6	GSC #7	GSC #8	GSC #9	GSC #10	GSC #11	GSC #12	GSC #13	GSC #14	AFP_{old}
130	4977	255	2	0	5	3	4	2	1	2	4	0	1	3	0	1	237
131	4067	219	0	0	2	3	2	0	2	0	5	0	3	3	0	4	195
132	7854	183	0	0	5	3	4	0	1	0	5	0	5	3	0	4	174
133	4172	206	3	0	4	3	4	4	3	2	2	0	3	3	0	1	200
134	9051	229	5	0	4	3	3	5	4	3	5	1	1	3	0	3	240
135	9520	583	5	1	3	3	3	5	4	3	4	0	1	3	0	5	612
136	2800	283	5	1	3	3	1	5	4	3	3	0	0	2	0	4	280
137	23940	1044	3	1	2	2	3	5	4	3	3	0	1	2	0	5	1034
138	1400	387	4	3	3	3	5	3	3	3	4	0	2	2	0	4	402
139	13860	662	5	1	3	3	3	5	4	5	4	0	1	3	0	3	695
140	5880	652	5	1	3	3	3	5	4	5	3	5	0	3	0	3	704
141	14973	516	2	0	3	3	3	2	3	2	5	1	2	4	0	4	511
142	4494	402	0	0	4	3	4	0	0	0	2	0	3	4	0	1	346
143	3437	349	0	0	3	3	3	0	0	0	2	0	2	4	0	3	297
144	5180	253	4	0	3	3	2	4	3	3	1	0	2	4	2	3	250
145	3192	101	5	0	3	3	2	5	4	4	1	2	4	4	3	3	109
146	840	99	4	0	3	3	3	3	4	0	0	0	2	4	2	1	93
147	5775	373	5	0	3	3	2	5	3	3	0	2	2	4	2	3	380
148	6405	292	5	0	2	0	1	5	3	3	1	1	3	3	5	3	292
149	19894	425	4	3	5	4	3	4	4	4	3	3	4	2	4	3	489
150	710	172	5	0	2	2	2	5	4	2	0	1	1	2	0	1	158
151	14987	464	4	0	1	1	2	0	3	0	3	0	3	2	0	0	390
152	651	182	5	0	3	3	3	5	5	4	3	1	3	3	0	0	187
153	2352	794	5	0	1	0	1	5	2	3	3	0	0	3	0	0	699
154	1435	384	5	0	2	0	2	5	2	4	2	1	0	2	1	2	357
155	8050	442	5	4	0	2	5	5	5	5	2	2	2	5	5	5	517
156	2429	241	4	2	0	3	0	5	4	3	2	3	3	3	5	4	255
157	4004	230	3	3	1	1	0	5	4	3	4	0	2	2	0	0	214
158	847	188	5	0	0	0	0	5	3	3	0	0	1	1	0	0	156
159	2174	119	1	1	3	3	3	0	4	0	3	1	1	2	0	3	107
160	9135	443	5	0	2	2	3	5	4	4	2	0	2	3	0	2	439
161	4620	506	5	0	2	2	1	5	3	4	0	0	0	3	0	3	471
162	6699	231	4	0	4	2	3	5	3	3	3	0	0	3	0	5	231
163	5922	411	5	2	2	0	0	5	3	4	1	0	0	2	0	0	366
164	4873	121	3	0	0	0	0	5	0	1	0	0	1	0	0	0	91
165	4200	196	3	1	1	0	1	5	3	1	0	0	1	0	0	2	163
166	5936	207	1	3	2	2	1	2	0	4	4	1	3	2	4	3	201
167	9583	193	5	0	3	2	3	5	4	4	4	0	2	2	1	2	197
168	7000	226	0	0	3	2	2	0	3	0	2	0	2	2	0	0	183
169	8876	229	4	1	5	2	3	5	3	4	0	0	2	2	0	0	220
170	10199	339	5	0	5	3	5	5	5	4	4	0	3	1	4	4	383
171	19285	406	3	0	2	2	2	5	4	4	3	2	2	2	0	2	398
172	2275	182	3	0	0	0	0	5	0	2	0	0	0	1	0	2	142

Project No	Effort (mh)	UFP	GSC #1	GSC #2	GSC #3	GSC #4	GSC #5	GSC #6	GSC #7	GSC #8	GSC #9	GSC #10	GSC #11	GSC #12	GSC #13	GSC #14	AFP$_{old}$
173	3941	277	5	0	2	1	0	5	2	2	2	0	2	1	0	0	241
174	9100	596	4	0	2	0	1	5	1	4	4	1	1	1	0	2	542
175	546	146	0	0	0	0	0	5	0	1	0	0	0	0	0	0	104
176	595	293	0	0	0	0	0	5	0	0	0	1	0	0	0	0	208
177	1155	146	5	0	0	0	0	5	0	0	0	0	0	0	0	0	110
178	5260	288	5	0	0	0	0	5	1	0	0	0	5	2	1	0	242
179	9800	640	5	0	5	3	0	5	5	0	5	3	3	3	0	3	672
180	15211	589	5	0	2	0	0	5	0	2	0	0	0	0	4	4	512
181	11235	514	3	0	3	0	1	5	2	1	5	0	1	3	0	2	468
182	2989	183	2	0	1	1	0	5	1	4	0	3	0	1	0	0	152
183	2331	104	0	0	1	1	0	0	1	0	2	0	0	1	0	0	74
184	14434	307	5	0	3	3	3	5	4	4	3	0	0	2	3	0	307
185	12824	318	5	4	2	5	3	5	4	3	2	3	3	4	4	4	369
186	5817	181	4	0	2	0	3	5	3	5	2	0	1	2	2	0	170
187	3136	124	4	3	2	1	2	5	4	4	2	0	1	2	2	0	120
188	2982	88	4	0	2	2	2	5	3	4	0	0	2	4	4	3	88
189	3514	122	5	0	3	2	2	5	3	4	1	0	1	4	3	4	124
190	2996	98	2	0	3	2	3	1	1	2	1	3	3	4	3	3	94
191	1981	256	5	0	3	2	1	5	3	4	0	0	2	3	2	2	248
192	7056	312	5	0	3	2	3	5	3	4	1	0	3	4	3	2	321
193	8512	151	5	0	3	2	3	5	2	4	1	0	3	4	3	3	156
194	4606	116	5	0	3	2	2	5	3	4	1	0	3	4	2	2	117
195	3500	102	0	0	3	2	3	0	1	0	3	1	3	4	3	4	94
196	5509	94	2	0	4	2	3	0	4	0	2	0	3	4	3	4	90
197	3472	253	2	0	1	0	0	5	0	1	2	0	0	1	0	3	202
198	1876	135	2	0	1	0	0	5	0	1	2	0	0	1	0	3	108
199	2926	240	4	0	2	0	0	5	2	1	3	0	1	1	0	4	211
200	3276	174	3	0	3	0	0	0	3	0	1	3	0	0	0	0	136
201	2723	165	2	0	1	0	0	5	0	1	2	0	0	1	0	2	130
202	3647	228	0	0	1	0	0	0	0	0	2	0	0	0	0	2	160
203	1575	74	2	0	1	0	0	5	0	1	2	0	0	1	0	2	58
204	2520	184	2	0	1	0	0	5	0	1	2	0	0	1	0	2	145
205	2583	146	3	0	2	0	0	4	2	1	3	0	0	1	0	2	121
206	1603	114	3	0	1	0	0	5	1	1	0	0	0	1	0	2	90
207	8232	439	3	0	1	0	0	5	0	1	3	0	0	1	0	2	356
208	6055	339	5	0	1	4	2	5	1	3	4	1	0	2	0	0	315
209	693	56	4	0	0	1	1	5	1	1	1	0	0	1	0	0	45
210	4438	223	3	3	0	0	0	5	0	2	1	0	0	2	0	0	181
211	889	53	0	0	2	3	2	0	4	0	3	0	1	2	0	0	43
212	3521	395	3	3	4	4	4	5	3	4	3	4	3	2	2	0	431
213	5922	456	4	0	2	2	0	5	0	1	1	3	0	0	0	0	378
214	1477	89	3	0	0	0	0	3	0	1	0	2	2	0	0	0	68
215	770	63	0	0	2	3	2	0	4	0	3	0	1	2	0	0	52

Project No	Effort (mh)	UFP	GSC #1	GSC #2	GSC #3	GSC #4	GSC #5	GSC #6	GSC #7	GSC #8	GSC #9	GSC #10	GSC #11	GSC #12	GSC #13	GSC #14	AFP_{old}
216	1141	101	0	0	2	3	2	0	4	0	3	0	1	2	0	0	83
217	2632	110	3	3	0	0	5	2	2	3	3	3	2	0	0	3	103
218	2303	104	4	3	4	0	0	5	0	1	1	0	2	0	1	1	90
219	1833	54	5	0	0	0	0	0	0	5	0	0	0	0	5	0	43
220	3596	405	3	0	0	0	0	5	0	0	3	3	2	0	4	0	344
221	2045	167	3	0	2	0	0	5	0	0	2	0	2	1	0	1	135
222	5267	263	5	0	0	0	0	5	0	3	3	5	1	0	4	3	247
223	5000	384	5	0	3	0	0	5	0	0	3	3	0	0	1	1	330
224	7974	258	5	0	5	0	0	1	3	3	3	0	5	0	5	0	245
225	3164	305	3	0	3	4	3	5	3	3	2	0	1	2	0	4	299
226	3948	423	4	0	3	4	3	5	3	3	2	0	4	2	0	4	431
227	4277	431	5	0	3	4	3	5	4	3	2	0	4	2	0	4	448
228	3542	301	5	0	4	3	2	5	2	3	1	3	0	2	3	4	307
229	3983	272	5	0	4	3	2	5	2	3	1	0	0	2	3	3	267
230	7252	281	2	0	3	4	3	1	3	3	2	0	0	2	0	4	259
231	3927	202	2	0	4	3	3	3	3	3	0	1	0	2	0	3	186
232	10577	371	4	0	2	4	2	5	4	4	3	1	3	3	0	4	386
233	86478	719	5	0	0	5	5	5	2	0	4	0	0	0	5	1	733
234	5859	215	5	0	5	0	0	5	2	5	0	3	0	0	3	4	219
235	5229	131	5	0	5	4	0	5	4	0	4	0	0	0	5	2	136
236	22498	272	2	0	0	4	0	0	5	3	0	0	0	3	5	2	256
237	2205	153	5	0	0	0	0	5	5	3	0	3	0	0	5	3	151
238	3626	250	2	0	3	3	3	5	3	3	5	1	3	4	0	1	252
239	1267	80	5	3	3	0	1	5	2	3	3	1	3	0	0	0	75
240	11361	514	5	3	3	0	3	4	3	4	1	2	4	2	0	3	524
241	2548	118	5	0	4	1	1	5	3	0	4	2	0	0	0	0	106
242	6783	311	5	0	3	1	2	5	1	2	0	2	5	1	0	1	289
243	4846	189	0	0	0	0	0	2	0	0	2	0	0	2	0	0	134
244	11218	92	0	0	0	0	0	0	0	0	2	2	0	2	0	0	65
245	7574	267	0	2	2	0	2	3	2	2	3	0	1	2	0	2	230
246	648	48	0	0	3	0	5	0	0	0	5	5	0	0	0	0	40
247	1674	87	0	0	0	0	0	1	0	0	2	0	0	0	0	0	59
248	3388	103	0	0	0	0	1	0	0	3	0	0	1	2	0	0	77
249	1971	66	0	0	1	0	0	0	0	0	0	0	1	1	0	0	45
250	5454	121	0	3	3	0	0	5	3	4	2	0	5	1	4	0	115
251	1647	115	0	0	0	0	0	2	0	0	2	0	0	2	0	0	82
252	4293	129	0	0	4	0	0	0	0	0	0	0	0	2	0	0	92
253	6331	170	0	0	0	0	0	0	0	0	2	0	0	2	0	0	117
254	23058	201	1	0	0	0	0	5	2	4	2	0	3	2	0	3	185
255	3591	145	0	0	0	0	0	0	0	0	2	0	0	2	0	0	100
256	8248	149	1	0	0	0	0	1	2	1	2	0	0	2	0	1	112
257	1418	67	0	0	0	0	0	0	0	0	2	5	0	2	0	0	51
258	12245	354	0	0	0	0	0	2	0	0	5	0	0	2	0	0	262

Project No	Effort (mh)	UFP	GSC #1	GSC #2	GSC #3	GSC #4	GSC #5	GSC #6	GSC #7	GSC #8	GSC #9	GSC #10	GSC #11	GSC #12	GSC #13	GSC #14	AFP_{old}
259	6412	201	0	0	0	0	0	0	0	0	2	0	0	2	0	0	141
260	2943	135	0	0	0	0	0	0	0	0	2	0	0	0	0	0	90
261	29970	694	4	3	5	3	5	5	5	4	5	2	5	5	0	3	861
262	16092	1257	0	0	5	0	2	0	0	0	4	0	0	2	2	0	1006
263	1633	83	1	0	0	0	0	0	0	0	2	0	0	3	0	0	59
264	5710	221	2	0	3	0	0	0	3	0	4	1	3	0	0	3	192
265	10570	281	3	0	2	1	3	0	3	3	1	1	2	0	3	3	261
266	67905	993	0	2	5	4	5	5	4	4	3	5	4	2	2	4	1182
267	1161	63	0	0	0	0	0	0	0	0	2	0	0	0	0	0	42
268	6318	381	4	0	5	0	0	0	0	2	0	2	0	0	0	1	301
269	1067	189	0	0	0	0	0	0	0	0	3	1	1	0	0	0	132
270	3267	173	0	0	3	3	0	0	0	0	0	1	3	1	0	3	140
271	5602	215	0	0	4	0	1	0	0	0	3	0	0	2	0	0	161
272	8802	517	0	0	0	0	3	0	3	0	2	0	0	0	0	0	377
273	17428	1045	1	1	2	2	2	4	4	4	5	0	3	0	0	2	993
274	4455	143	0	0	0	0	0	0	0	0	2	0	0	2	0	0	99
275	9342	288	3	0	3	0	2	2	1	2	2	2	0	1	0	4	256
276	2295	245	5	0	5	4	4	5	5	3	0	0	5	0	0	4	257
277	14188	565	5	3	4	4	0	1	2	0	2	1	3	3	3	5	599
278	4226	189	0	1	0	2	0	0	0	0	2	1	0	1	0	0	136
279	675	92	0	0	0	0	0	0	0	0	5	0	0	0	0	3	67
280	2498	56	0	0	0	0	0	2	0	0	3	0	0	2	0	0	40
281	2862	144	5	0	5	0	0	0	0	0	1	0	3	0	0	2	117
282	1472	104	3	2	0	0	0	0	0	0	0	0	0	0	0	0	73
283	7304	223	0	5	0	0	0	2	4	2	2	0	0	0	0	0	178
284	6183	157	0	0	5	0	3	4	4	5	5	2	2	0	0	2	152
285	4738	1043	0	0	0	0	0	0	0	0	0	0	0	0	0	0	678
286	1601	190	0	0	0	0	0	0	0	0	0	0	0	0	0	0	124
287	598	149	0	0	0	0	0	0	0	0	0	0	0	0	0	0	97
288	2124	417	0	0	0	0	0	0	0	0	0	0	0	0	0	0	271
289	8165	1206	0	0	0	0	0	0	0	0	0	0	0	0	0	0	784
290	2255	174	0	0	0	0	0	0	0	0	0	0	0	0	0	0	113
291	4956	669	0	0	0	0	0	0	0	0	0	0	0	0	0	0	435
292	1638	269	0	0	0	0	0	0	0	0	0	0	0	0	0	0	175
293	2975	569	0	0	0	0	0	0	0	0	0	0	0	0	0	0	370
294	1596	302	0	0	0	0	0	0	0	0	0	0	0	0	0	0	196
295	3084	373	0	0	0	0	0	0	0	0	0	0	0	0	0	0	242
296	1744	322	0	0	0	0	0	0	0	0	0	0	0	0	0	0	209
297	1154	173	0	0	0	0	0	0	0	0	0	0	0	0	0	0	112
298	5806	196	0	0	0	0	0	0	0	0	0	0	0	0	0	0	127
299	1103	225	0	0	0	0	0	0	0	0	0	0	0	0	0	0	146
300	21143	1110	4	3	4	2	4	1	3	3	3	1	1	3	0	4	21354
301	10930	391	4	3	3	2	3	1	3	3	3	1	1	3	0	4	10821

APPENDIX C

PROJECT DATA USED IN REGRESSION ANALYSIS TO DETERMINE MULTIPLIERS (n_i)

Project No.	Effort (mh)	UFP	GSC(adj)1	GSC(adj)2	GSC(adj)3	GSC(adj)4	GSC(adj)5	GSC(adj)6	GSC(adj)7	GSC(adj)8	GSC(adj)9	GSC(adj)10	GSC(adj)11	GSC(adj)12	GSC(adj)13	GSC(adj)14	LOG(Effort)	LOG(UFP)
2	2569	168	2	-3	-1	0	-2	2	-1	0	-3	-3	0	-2	-3	-3	3.4097641	2.22530928
3	805	107	2	-3	-1	0	-3	-3	2	-3	-3	-1	-3	-2	-3	-3	2.90579588	2.02938378
5	3829	296	2	-3	-2	-2	-2	2	2	0	0	-1	1	-3	-3	-3	3.58308537	2.47129171
6	5635	328	2	-3	-1	-2	-2	2	1	1	-1	1	-2	-3	-3	1	3.75089392	2.51587384
8	2821	193	1	-3	2	-2	2	2	2	0	-1	1	-1	-3	-3	1	3.45040309	2.28555731
11	4908	320	-3	-3	-2	-3	-2	-2	-2	-3	0	-2	0	0	-3	-3	3.69090455	2.50514998
12	7721	254	2	-3	-2	-3	-2	2	2	0	2	-2	2	-3	-3	2	3.88767355	2.40483372
13	18314	375	2	-3	2	2	0	2	0	0	2	-3	2	0	-3	2	4.26278321	2.57403127
15	4273	99	-3	-3	2	-3	0	-3	0	-3	2	-2	0	-2	-3	-3	3.63073289	1.99563519
17	5470	364	-3	-3	2	-3	0	-3	0	-3	2	-2	0	-2	-3	-3	3.73798733	2.56110138
18	2155	75	2	-3	2	-3	2	2	0	2	2	-2	-2	-2	-3	-3	3.33344727	1.87506126
21	5985	269	0	-3	0	-3	0	2	-2	0	-2	-2	-2	0	-3	0	3.77706415	2.42975228
22	1784	175	2	-3	-2	-3	0	2	0	0	0	-2	-2	-2	-3	0	3.25139485	2.24303805
24	2514	244	-2	-3	2	-3	2	0	-2	-3	2	-2	0	0	-3	-3	3.40036527	2.38738983
25	501	186	0	-3	-3	-3	-3	2	-3	-3	-3	-3	-3	-3	-3	-3	2.69983773	2.26951294
29	14700	348	2	-3	1	0	2	2	-1	0	-3	1	0	-3	-3	-3	4.16731733	2.54157924
31	1050	337	2	-3	2	-3	1	2	1	2	-3	0	1	0	-2	-3	3.0211893	2.5276299
32	3000	560	2	-3	-3	-3	-3	2	-3	-3	-3	-3	-2	-3	-3	-3	3.47712125	2.74818803
34	6557	367	2	-3	-1	-3	-3	2	0	-3	-2	-1	0	-3	-3	-2	3.81670518	2.56466606
35	5865	560	2	-3	0	-3	-3	2	1	-3	0	-3	-3	-3	-2	-3	3.76826802	2.74818803
37	12630	1003	2	-3	2	-3	2	2	-3	1	1	0	0	-3	-3	0	4.10140335	3.00130093
40	5568	328	2	-3	-3	-1	-1	2	-1	-3	-2	-3	-2	-2	-3	-2	3.74569923	2.51587384
41	6129	397	2	-3	-2	-2	-2	2	-3	-3	-2	-3	-3	-3	-2	-3	3.78738962	2.59879051
42	247	172	2	-2	-3	-3	-3	2	-3	-3	-2	-3	0	-3	-3	-3	2.39269695	2.23552845
43	4291	681	2	-3	-3	-3	2	2	0	2	-3	-1	1	-3	0	-3	3.63255851	2.83314711
45	60142	851	0	1	1	2	1	2	2	2	2	0	-3	-3	-3	2	4.77917787	2.92992956
46	4874	228	-3	-3	0	-3	-3	-2	-2	-3	0	-3	0	-3	-3	-3	3.68788552	2.35793485
50	1944	82	-3	-3	-3	-3	-3	-3	-3	-3	-1	-3	-3	-3	-3	-3	3.28869626	1.91381385
51	1526	84	-3	-3	-3	-3	-3	-3	-3	-3	-1	-3	-3	-1	-3	-3	3.18355453	1.92427929
53	4320	147	-3	-3	-1	-3	-3	-3	-3	-3	0	-2	-3	-1	-3	2	3.63548375	2.16731733
56	4792	139	-3	-3	-3	-3	-3	-3	-3	-3	-1	0	0	-3	-3	-3	3.68051681	2.1430148
57	1256	108	-3	-3	-3	-3	-1	-3	-3	-3	-1	-3	-3	-3	-3	-3	3.09898964	2.03342376
58	12164	208	-3	-3	0	-3	-1	0	0	-1	0	-1	-2	-3	-3	-2	4.08507641	2.31806333
59	4658	261	-3	-3	-2	-3	-3	-3	-3	-3	-1	-3	-3	-1	-3	-1	3.66819946	2.41664051
60	5926	196	-3	-3	-3	-3	-3	-3	-3	-3	-1	-1	-2	-3	-3	0	3.77276165	2.29225607
63	7654	431	-3	-3	-3	-3	-3	-2	-3	-3	0	-3	-3	-3	-3	-2	3.88388846	2.63447727
64	41445	669	0	-3	1	-2	0	0	1	-2	2	-3	0	0	-2	0	4.61747214	2.82542612
65	2160	149	1	0	-1	0	-3	-2	1	-3	-2	-3	-3	-3	-3	0	3.33445375	2.17318627
66	9801	204	1	1	-3	-3	-3	-3	-3	-3	0	-3	-2	-1	-3	-3	3.99127039	2.30963017
68	8235	167	-3	-3	-2	-3	-2	-3	-3	-3	2	-3	-3	-2	-3	-3	3.9156636	2.22271647
69	4050	155	-3	-3	-3	-3	-3	-3	-3	-3	-1	-3	-2	-1	-3	-3	3.60745502	2.1903317
72	7614	460	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	3.88161287	2.66275783
74	15916	1003	2	-1	-2	-3	-3	-3	-3	-3	1	-2	0	-1	-2	-2	4.20183393	3.00130093
75	5386	477	1	0	0	-3	-3	-3	-3	-3	-1	-1	-2	-2	-3	-1	3.73126635	2.67851838
78	1904	126	-3	-3	-3	-3	-2	-3	-3	-3	2	-3	-1	-1	-3	0	3.27966694	2.10037055
79	8883	237	0	-3	-3	-3	-3	-3	-3	-3	0	-3	-3	-1	-3	-1	3.94855966	2.37474835
82	7155	179	-3	-3	0	-3	-3	-1	0	-1	1	0	-1	-1	-1	0	3.85460964	2.25285303
83	10962	352	-3	-3	0	-3	-3	-2	-3	-3	1	-3	0	-3	-3	-3	4.0398896	2.54654266
84	7169	419	-3	-3	-1	-3	-3	-3	-3	-3	-1	-3	-3	-3	-3	-3	3.85545858	2.62221402
85	6251	264	-2	-3	-3	-3	-3	-3	-3	-3	0	-3	-3	-3	-3	-3	3.7959495	2.42160393
86	8249	306	1	-3	-3	-3	-3	0	-1	-2	-1	-3	-2	0	-3	-2	3.9164013	2.48572143
87	7601	347	-3	-3	-3	-3	-3	-3	-3	-3	2	-3	-3	-3	-3	2	3.88087073	2.54032947

Project No.	Effort (mh)	UFP	GSC(adj)1	GSC(adj)2	GSC(adj)3	GSC(adj)4	GSC(adj)5	GSC(adj)6	GSC(adj)7	GSC(adj)8	GSC(adj)9	GSC(adj)10	GSC(adj)11	GSC(adj)12	GSC(adj)13	GSC(adj)14	LOG(Effort)	LOG(UFP)
88	7074	239	-3	-3	2	-3	-3	-3	-3	-3	2	-3	-3	-1	-3	0	3.84966506	2.3783979
89	4536	174	0	-2	-3	-3	-2	-2	0	-2	-3	-3	-2	-3	-1	-1	3.65667305	2.24054925
90	5049	477	0	-3	-1	-3	0	2	2	1	-3	-3	-3	-1	-2	-1	3.70320537	2.67851838
91	9315	655	0	-3	1	0	1	1	1	1	1	-3	0	-3	-3	-3	3.96918286	2.8162413
92	13703	134	2	2	2	0	-2	-2	1	-3	2	2	0	-3	-3	2	4.13681566	2.1271048
93	6494	292	0	-3	2	-3	1	2	0	1	2	-1	1	0	-3	-1	3.81251228	2.46538285
95	6926	331	-3	-3	-3	-3	-3	-2	-3	-3	-2	-3	-3	-2	0	-3	3.84048249	2.51982799
96	54716	820	1	-3	2	-3	1	2	1	1	2	2	2	-1	-3	1	4.73811434	2.91381385
97	3942	129	-3	-3	-3	-3	-3	-3	-3	-3	-1	-3	-3	-1	-1	-2	3.59571662	2.11058971
98	4293	341	0	0	2	-2	-3	2	1	1	-1	-3	-3	-1	0	1	3.63276089	2.53275438
99	7519	676	-2	-3	-3	0	0	2	2	1	-3	-3	-3	-3	-3	0	3.87616008	2.8299467
100	2565	153	-3	-3	-3	-3	-3	-3	-3	-3	-1	-3	-3	-1	-3	-3	3.40906737	2.18469143
102	16996	425	-3	-3	0	-3	-3	-3	-3	-3	1	-1	1	0	-3	0	4.23034672	2.62838893
103	12271	363	-3	-3	1	-3	-3	-3	-3	-3	2	0	-1	-1	-3	-3	4.08887998	2.55990663
104	7884	326	0	-3	2	-3	1	2	0	1	2	-1	1	0	-3	-1	3.89674662	2.5132176
105	4509	168	-3	-3	-3	-3	-3	-3	-3	-3	-1	-3	-2	-3	-3	-3	3.65408024	2.22530928
106	2254	78	-3	-3	-3	-3	-3	-3	-3	-3	-1	-3	-3	-3	-3	-3	3.35295391	1.8920946
107	15309	590	2	1	2	2	2	2	2	1	0	-3	2	-1	0	2	4.18494682	2.77085201
108	4860	277	2	-3	2	2	2	0	0	1	0	2	2	-3	-3	2	3.68663627	2.44247977
109	1566	72	-3	-3	-3	-3	-3	-3	-3	-3	1	-3	-3	-3	-3	0	3.19479176	1.8573325
110	26474	820	2	-3	2	-1	2	2	1	1	0	-3	2	0	-3	2	4.42281956	2.91381385
111	1512	85	-2	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	3.17955179	1.92941893
113	3132	134	1	1	-1	0	-3	-3	-3	-3	-3	-3	0	-1	1	0	3.49582175	2.1271048
115	8006	220	-3	-3	-3	-3	-3	-3	-3	-3	1	-3	-3	-1	-3	-3	3.90341559	2.34242268
116	6656	479	-3	-3	-3	-3	-3	-3	-3	-3	0	-3	-3	-1	-3	-1	3.82321331	2.68033551
118	3874	173	-3	-3	-3	-3	-3	-3	-3	-3	-1	-3	-3	-1	-3	-3	3.58815962	2.2380461
119	11003	961	1	-1	-1	-3	-2	2	-1	-1	1	-3	1	-2	-3	0	4.04151111	2.98272339
120	3992	280	2	-3	-2	-3	-3	2	-2	-3	-3	-3	-3	-3	-3	-2	3.60119053	2.44715803
121	20205	568	2	-3	0	-3	-1	2	1	-3	-3	-1	-1	-3	-3	1	4.30545885	2.75434834
122	493	100	2	-3	-3	-3	-3	2	-3	-3	-2	-2	-2	-3	-3	-3	2.69284692	2
123	2150	176	2	-3	-3	-3	-3	2	-3	-3	-2	-2	-2	-3	-3	-3	3.33243846	2.24551267
124	3452	188	2	-3	-3	-3	-3	2	-3	-3	-2	-2	-2	-3	-3	-3	3.53807079	2.27415785
125	5522	270	2	-1	-3	-3	-3	2	-3	-3	-3	-3	-3	-3	-3	1	3.7420964	2.43136376
127	1617	162	0	-3	-1	0	0	2	-1	-1	-3	-1	-2	0	-3	-3	3.20871002	2.20951501
128	2422	123	0	-3	-1	0	0	2	-2	0	-3	-3	-2	0	-3	-3	3.38417414	2.08990511
129	2282	112	-3	-3	-1	0	-1	-3	-2	-3	2	-3	-2	0	-3	-1	3.35831564	2.04921802
130	4977	255	-1	-3	2	0	1	-1	-2	-1	1	-3	-2	0	-3	-2	3.69696764	2.40654018
131	4067	219	-3	-3	-1	0	-1	-3	-1	-3	2	-3	0	0	-3	1	3.60927417	2.34044411
132	7854	183	-3	-3	2	0	1	-3	-2	-3	2	-3	2	0	-3	1	3.8950909	2.26245109
134	9051	229	2	-3	1	0	0	2	1	0	2	-2	-2	0	-3	0	3.95669656	2.35983548
135	9520	583	2	-2	0	0	0	2	1	0	1	-3	-2	0	-3	2	3.97863695	2.76566855
137	23940	1044	0	-2	-1	-1	0	2	1	0	0	-3	-2	-1	-3	2	4.37912415	3.0187005
138	1400	387	1	0	0	0	2	0	0	0	1	-3	-1	-1	-3	1	3.14612804	2.58771097
139	13860	662	2	-2	0	0	0	2	1	2	1	-3	-2	0	-3	0	4.14176323	2.82085799
140	5880	652	2	-2	0	0	0	2	1	2	0	2	-3	0	-3	0	3.76937733	2.8142476
142	4494	402	-3	-3	1	0	1	-3	-3	-3	-1	-3	0	1	-3	-2	3.65263307	2.60422605
143	3437	349	-3	-3	0	0	0	-3	-3	-3	-1	-3	-1	1	-3	0	3.53617953	2.54282543
144	5180	253	1	-3	0	0	-1	1	0	0	-2	-3	-1	1	-1	0	3.71432976	2.40312052
146	840	99	1	-3	0	0	0	0	1	-3	-3	-3	-1	1	-1	-2	2.92427929	1.99563519
148	6405	292	2	-3	-1	-3	-2	2	0	0	-2	-2	0	0	2	0	3.80651913	2.46538285
149	19894	425	1	0	2	1	0	1	1	1	0	0	1	-1	1	0	4.29872211	2.62838893
152	651	182	2	-3	0	0	0	2	2	1	0	-2	0	0	-3	-3	2.81358099	2.26007139

Project No.	Effort (mh)	UFP	GSC(adj)1	GSC(adj)2	GSC(adj)3	GSC(adj)4	GSC(adj)5	GSC(adj)6	GSC(adj)7	GSC(adj)8	GSC(adj)9	GSC(adj)10	GSC(adj)11	GSC(adj)12	GSC(adj)13	GSC(adj)14	LOG(Effort)	LOG(UFP)
154	1435	384	2	-3	-1	-3	-1	2	-1	1	-1	-2	-3	-1	-2	-1	3.1568519	2.58433122
155	8050	442	2	1	-3	-1	2	2	2	2	-1	-1	-1	2	2	2	3.90579588	2.64542227
156	2429	241	1	-1	-3	0	-3	2	1	0	-1	0	0	0	2	1	3.38542751	2.38201704
157	4004	230	0	0	-2	-2	-3	2	1	0	1	-3	-1	-1	-3	-3	3.60249407	2.36172784
158	847	188	2	-3	-3	-3	-3	2	0	0	-3	-3	-2	-2	-3	-3	2.92788341	2.27415785
159	2174	119	-2	-2	0	0	0	-3	1	-3	0	-2	-2	-1	-3	0	3.33725954	2.07554696
162	6699	231	1	-3	1	-1	0	2	0	0	0	-3	-3	0	-3	2	3.82600998	2.36361198
164	4873	121	0	-3	-3	-3	-3	2	-3	-2	-3	-3	-2	-3	-3	-3	3.68779641	2.08278537
165	4200	196	0	-2	-2	-3	-2	2	0	-2	-3	-3	-2	-3	-3	-1	3.62324929	2.29225607
166	5936	207	-2	0	-1	-1	-2	-1	-3	1	1	-2	0	-1	1	0	3.77349389	2.31597035
167	9583	193	2	-3	0	-1	0	2	1	1	1	-3	-1	-1	-2	-1	3.98150149	2.28555731
170	10199	339	2	-3	2	0	2	2	2	1	1	-3	0	-2	1	1	4.00855759	2.5301997
171	19285	406	0	-3	-1	-1	-1	2	1	1	0	-1	-1	-1	-3	-1	4.28521984	2.60852603
173	3941	277	2	-3	-1	-2	-3	2	-1	-1	-1	-3	-1	-2	-3	-3	3.59560643	2.44247977
175	546	146	-3	-3	-3	-3	-3	2	-3	-2	-3	-3	-3	-3	-3	-3	2.73719264	2.16435286
177	1155	146	2	-3	-3	-3	-3	2	-3	-3	-3	-3	-3	-3	-3	-3	3.06258198	2.16435286
178	5260	288	2	-3	-3	-3	-3	2	-2	-3	-3	-3	2	-1	-2	-3	3.72098574	2.45939249
179	9800	640	2	-3	2	0	-3	2	2	-3	2	0	0	0	-3	0	3.99122608	2.80617997
180	15211	589	2	-3	-1	-3	-3	2	-3	-1	-3	-3	-3	-3	1	1	4.18215777	2.77011529
181	11235	514	0	-3	0	-3	-2	2	-1	-2	2	-3	-2	0	-3	-1	4.05057308	2.71096312
183	2331	104	-3	-3	-2	-2	-3	-3	-2	-3	-1	-3	-3	-2	-3	-3	3.36754227	2.01703334
184	14434	307	2	-3	0	0	0	2	1	1	0	-3	-3	-1	0	-3	4.1593867	2.48713838
186	5817	181	1	-3	-1	-3	0	2	0	2	-1	-3	-2	-1	-1	-3	3.76469906	2.25767857
187	3136	124	1	0	-1	-2	-1	2	1	1	-1	-3	-2	-1	-1	-3	3.49637605	2.09342169
188	2982	88	1	-3	-1	-1	-1	2	0	1	-3	-3	-1	1	1	0	3.47450764	1.94448267
189	3514	122	2	-3	0	-1	-1	2	0	1	-2	-3	-2	1	0	1	3.54580176	2.08635983
190	2996	98	-1	-3	0	-1	0	-2	-2	-1	-2	0	0	1	0	0	3.47654181	1.99122608
191	1981	256	2	-3	0	-1	-2	2	0	1	-3	-3	-1	0	-1	-1	3.29688448	2.40823997
192	7056	312	2	-3	0	-1	0	2	0	1	-2	-3	0	1	0	-1	3.84855857	2.49415459
194	4606	116	2	-3	0	-1	-1	2	0	1	-2	-3	0	1	-1	-1	3.66332393	2.06445799
195	3500	102	-3	-3	0	-1	0	-3	-2	-3	0	-2	0	1	0	1	3.54406804	2.00860017
196	5509	94	-1	-3	1	-1	0	-3	1	-3	-1	-3	0	1	0	1	3.74107277	1.97312785
197	3472	253	-1	-3	-2	-3	-3	2	-3	-2	-1	-3	-3	-2	-3	0	3.54057972	2.40312052
199	2926	240	1	-3	-1	-3	-3	2	-1	-2	0	-3	-2	-2	-3	1	3.46627432	2.38021124
200	3276	174	0	-3	0	-3	-3	-3	0	-3	-2	0	-3	-3	-3	-3	3.51534389	2.24054925
202	3647	228	-3	-3	-2	-3	-3	-3	-3	-3	-1	-3	-3	-3	-3	-1	3.56193576	2.35793485
203	1575	74	-1	-3	-2	-3	-3	2	-3	-2	-1	-3	-3	-2	-3	-1	3.19728056	1.86923172
204	2520	184	-1	-3	-2	-3	-3	2	-3	-2	-1	-3	-3	-2	-3	-1	3.40140054	2.26481782
206	1603	114	0	-3	-2	-3	-3	2	-2	-2	-3	-3	-3	-2	-3	-1	3.20493352	2.05690485
207	8232	439	0	-3	-2	-3	-3	2	-3	-2	0	-3	-3	-2	-3	-1	3.91550536	2.64246452
208	6055	339	2	-3	-2	1	-1	2	-2	0	1	-2	-3	-1	-3	-3	3.78211415	2.5301997
209	693	56	1	-3	-3	-2	-2	2	-2	-2	-2	-3	-3	-2	-3	-3	2.84073323	1.74818803
210	4438	223	0	0	-3	-3	-3	2	-3	-1	-2	-3	-3	-1	-3	-3	3.6471873	2.34830486
211	889	53	-3	-3	-1	0	-1	-3	1	-3	0	-3	-2	-1	-3	-3	2.94890176	1.72427587
213	5922	456	1	-3	-1	-1	-3	2	-3	-2	-2	0	-3	-3	-3	-3	3.7724684	2.65896484
214	1477	89	0	-3	-3	-3	-3	0	-3	-2	-3	-1	-1	-3	-3	-3	3.1693805	1.94939001
215	770	63	-3	-3	-1	0	-1	-3	1	-3	0	-3	-2	-1	-3	-3	2.88649073	1.79934055
216	1141	101	-3	-3	-1	0	-1	-3	1	-3	0	-3	-2	-1	-3	-3	3.05728564	2.00432137
219	1833	54	2	-3	-3	-3	-3	-3	-3	2	-3	-3	-3	-3	2	-3	3.26316246	1.73239376
220	3596	405	0	-3	-3	-3	-3	2	-3	-3	0	0	-1	-3	1	-3	3.55581988	2.60745502
221	2045	167	0	-3	-1	-3	-3	2	-3	-3	-1	-3	-1	-2	-3	-2	3.31069331	2.22271647
222	5267	263	2	-3	-3	-3	-3	2	-3	0	0	2	-2	-3	1	0	3.72156332	2.41995575

Project No.	Effort (mh)	UFP	GSC(adj)1	GSC(adj)2	GSC(adj)3	GSC(adj)4	GSC(adj)5	GSC(adj)6	GSC(adj)7	GSC(adj)8	GSC(adj)9	GSC(adj)10	GSC(adj)11	GSC(adj)12	GSC(adj)13	GSC(adj)14	LOG(Effort)	LOG(UFP)
223	5000	384	2	-3	0	-3	-3	2	-3	-3	0	0	-3	-3	-2	-2	3.69897	2.58433122
224	7974	258	2	-3	2	-3	-3	-2	0	0	0	-3	2	-3	2	-3	3.90167623	2.41161971
226	3948	423	1	-3	0	1	0	2	0	0	-1	-3	1	-1	-3	1	3.59637714	2.62634037
227	4277	431	2	-3	0	1	0	2	1	0	-1	-3	1	-1	-3	1	3.63113925	2.63447727
229	3983	272	2	-3	1	0	-1	2	-1	0	-2	-3	-3	-1	0	0	3.60021031	2.4345689
233	86478	719	2	-3	-3	2	2	2	-1	-3	1	-3	-3	-3	2	-2	4.93690564	2.85672889
234	5859	215	2	-3	2	-3	-3	2	-1	2	-3	0	-3	-3	0	1	3.7678235	2.33243846
235	5229	131	2	-3	2	1	-3	2	1	-3	1	-3	-3	-3	2	-1	3.71841864	2.1172713
236	22498	272	-1	-3	-3	1	-3	-3	2	0	-3	-3	-3	0	2	-1	4.35214391	2.4345689
237	2205	153	2	-3	-3	-3	-3	2	2	0	-3	0	-3	-3	2	0	3.34340659	2.18469143
238	3626	250	-1	-3	0	0	0	2	0	0	2	-2	0	1	-3	-2	3.5594278	2.39794001
241	2548	118	2	-3	1	-2	-2	2	0	-3	1	-1	-3	-3	-3	-3	3.40619942	2.07188201
242	6783	311	2	-3	0	-2	-1	2	-2	-1	-3	-1	2	-2	-3	-2	3.83142182	2.49276039
243	4846	189	-3	-3	-3	-3	-3	-1	-3	-3	-1	-3	-3	-1	-3	-3	3.68538341	2.2764618
244	11218	92	-3	-3	-3	-3	-3	-3	-3	-3	-1	-1	-3	-1	-3	-3	4.04991544	1.96378783
245	7574	267	-3	-1	-1	-3	-1	0	-1	-1	0	-3	-2	-1	-3	-1	3.8793253	2.42651126
246	648	48	-3	-3	0	-3	2	-3	-3	-3	2	2	-3	-3	-3	-3	2.81157501	1.68124124
248	3388	103	-3	-3	-3	-3	-2	-3	-3	0	-3	-3	-2	-1	-3	-3	3.5299434	2.01283722
249	1971	66	-3	-3	-2	-3	-3	-3	-3	-3	-3	-3	-2	-2	-3	-3	3.29468662	1.81954394
250	5454	121	-3	0	0	-3	-3	2	0	1	-1	-3	2	-2	1	-3	3.73671513	2.08278537
251	1647	115	-3	-3	-3	-3	-3	-1	-3	-3	-1	-3	-3	-1	-3	-3	3.2166936	2.06069784
252	4293	129	-3	-3	1	-3	-3	-3	-3	-3	-3	-3	-3	-1	-3	-3	3.63276089	2.11058971
253	6331	170	-3	-3	-3	-3	-3	-3	-3	-3	-1	-3	-3	-1	-3	-3	3.80147231	2.23044892
254	23058	201	-2	-3	-3	-3	-3	2	-1	1	-1	-3	0	-1	-3	0	4.36282163	2.30319606
255	3591	145	-3	-3	-3	-3	-3	-3	-3	-3	-1	-3	-3	-1	-3	-3	3.55521541	2.161368
258	12245	354	-3	-3	-3	-3	-3	-1	-3	-3	2	-3	-3	-1	-3	-3	4.08795879	2.54900326
260	2943	135	-3	-3	-3	-3	-3	-3	-3	-3	-1	-3	-3	-3	-3	-3	3.46879026	2.13033377
261	29970	694	1	0	2	0	2	2	2	1	2	-1	2	2	-3	0	4.47666674	2.84135947
263	1633	83	-2	-3	-3	-3	-3	-3	-3	-3	-1	-3	-3	0	-3	-3	3.21298618	1.91907809
265	10570	281	0	-3	-1	-2	0	-3	0	0	-2	-2	-1	-3	0	0	4.02407499	2.44870632
266	67905	993	-3	-1	2	1	2	2	1	1	0	2	1	-1	-1	1	4.63190175	2.99694925
268	6318	381	1	-3	2	-3	-3	-3	-3	-1	-3	-1	-3	-3	-3	-2	3.80057962	2.58092498
271	5602	215	-3	-3	1	-3	-2	-3	-3	-3	0	-3	-3	-1	-3	-3	3.7483431	2.33243846
272	8802	517	-3	-3	-3	-3	0	-3	0	-3	-1	-3	-3	-3	-3	-3	3.94458136	2.71349054
273	17428	1045	-2	-2	-1	-1	-1	1	1	1	2	-3	0	-3	-3	-1	4.24124755	3.01911629
274	4455	143	-3	-3	-3	-3	-3	-3	-3	-3	-1	-3	-3	-1	-3	-3	3.64884771	2.15533604
275	9342	288	0	-3	0	-3	-1	-1	-2	-1	-1	-1	-3	-2	-3	1	3.97043986	2.45939249
276	2295	245	2	-3	2	1	1	2	2	0	-3	-3	2	-3	-3	1	3.36078269	2.38916608
277	14188	565	2	0	1	1	-3	-2	-1	-3	-1	-2	0	0	0	2	4.15192118	2.75204845
278	4226	189	-3	-2	-3	-1	-3	-3	-3	-3	-1	-2	-3	-2	-3	-3	3.62592949	2.2764618
279	675	92	-3	-3	-3	-3	-3	-3	-3	-3	2	-3	-3	-3	-3	0	2.82930377	1.96378783
280	2498	56	-3	-3	-3	-3	-3	-1	-3	-3	0	-3	-3	-1	-3	-3	3.39759243	1.74818803
281	2862	144	2	-3	2	-3	-3	-3	-3	-3	-2	-3	0	-3	-3	-1	3.45666963	2.15836249
282	1472	104	0	-1	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	3.16790781	2.01703334
284	6183	157	-3	-3	2	-3	0	1	1	2	2	-1	-1	-3	-3	-1	3.79119925	2.19589965
285	4738	1043	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	3.67559506	3.01828431
287	598	149	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	2.77670118	2.17318627
290	2255	174	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	3.35314655	2.24054925
292	1638	269	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	3.2143139	2.42975228
293	2975	569	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	3.47348697	2.75511227
297	1154	173	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	3.06220581	2.2380461
299	1103	225	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	3.04257551	2.35218252

Project No.	Effort (mh)	UFP	GSC(adj)1	GSC(adj)2	GSC(adj)3	GSC(adj)4	GSC(adj)5	GSC(adj)6	GSC(adj)7	GSC(adj)8	GSC(adj)9	GSC(adj)10	GSC(adj)11	GSC(adj)12	GSC(adj)13	GSC(adj)14	LOG(Effort)	LOG(UFP)
300	21143	1110	1	0	1	-1	1	-2	0	0	0	-2	-2	0	-3	1	4.32516661	3.04532298
301	10930	391	1	0	0	-1	0	-2	0	0	0	-2	-2	0	-3	1	4.03862016	2.59217676

SUMMARY OUTPUT										
Regression Statistics										
Multiple R	0.490486206									
R Square	0.240576719									
Adjusted R Square	0.186054021									
Standard Error	0.27809023									
Observations	210									
ANOVA										
	df	*SS*	*MS*	*F*	*Significance F*					
Regression	14	4.777225752	0.341230411	4.412414118	7.79417E-07					
Residual	195	15.08016436	0.077334176							
Total	209	19.85739011								
	Coefficients	*Standard Error*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*	*Lower 95.000%*	*Upper 95.000%*		ni
Intercept	1.520152257	0.06872148	22.120482	4.66863E-55	1.384619557	1.655684958	1.384619557	1.655684958		
GSC(adj)1	-0.022334825	0.014551987	-1.534829906	0.126446505	-0.051034296	0.006364646	-0.051034296	0.006364646		0.949872
GSC(adj)2	0.000453525	0.020330894	0.022307206	0.982225711	-0.039643122	0.040550172	-0.039643122	0.040550172		1.001045
GSC(adj)3	0.016112868	0.015329247	1.051119353	0.294504631	-0.014119517	0.046345254	-0.014119517	0.046345254		1.037798
GSC(adj)4	0.005114788	0.018321394	0.279170257	0.780410028	-0.031018718	0.041248295	-0.031018718	0.041248295		1.011847
GSC(adj)5	-0.024735588	0.017940593	-1.378749724	0.169551484	-0.060118078	0.010646902	-0.060118078	0.010646902		0.944636
GSC(adj)6	-0.017787061	0.013618664	-1.306079738	0.193064003	-0.044645828	0.009071707	-0.044645828	0.009071707		0.959871
GSC(adj)7	6.56293E-05	0.016223005	0.004045449	0.996776343	-0.031929427	0.032060686	-0.031929427	0.032060686		1.000151
GSC(adj)8	0.020892928	0.017873785	1.168914564	0.243865006	-0.014357803	0.056143658	-0.014357803	0.056143658		1.049284
GSC(adj)9	0.052255828	0.013791721	3.788927346	0.000201521	0.025055756	0.079455899	0.025055756	0.079455899		1.127862
GSC(adj)10	0.014606721	0.01675675	0.871691738	0.384448536	-0.01844099	0.047654431	-0.01844099	0.047654431		1.034205
GSC(adj)11	-0.00127279	0.015056884	-0.084532093	0.932720109	-0.03096802	0.02842244	-0.03096802	0.02842244		0.997074
GSC(adj)12	0.011194433	0.017052901	0.656453265	0.512306434	-0.022437348	0.044826213	-0.022437348	0.044826213		1.026111
GSC(adj)13	0.050561433	0.015320774	3.300187795	0.001148791	0.020345757	0.080777108	0.020345757	0.080777108		1.12347
GSC(adj)14	0.018980474	0.013833476	1.372068296	0.171618522	-0.008301947	0.046262895	-0.008301947	0.046262895		1.044673

APPENDIX D

DATA FOR 91 PROJECTS USED TO TEST PROPOSED APPROACH

Project No.	Effort(mh)	UFP	AFP$_{old}$	GSC(adj)1	GSC(adj)2	GSC(adj)3	GSC(adj)4	GSC(adj)5	GSC(adj)6	GSC(adj)7	GSC(adj)8	GSC(adj)9	GSC(adj)10	GSC(adj)11	GSC(adj)12	GSC(adj)13	GSC(adj)14	Overall Multiplier	AFP$_{new}$
1	5152	204	202	2	-3	0	0	-1	2	1	0	-3	1	0	-2	-3	-2	0.388151317	79
4	3913	227	204	1	-3	-1	0	-3	2	-2	1	1	-3	-2	-2	-3	-3	0.630307778	143
7	2149	241	214	2	-2	-2	-2	-2	-3	1	-1	0	0	-3	-3	-3	-1	0.566690118	144
9	3040	179	168	2	-3	-2	-3	0	2	0	0	0	-2	-2	-2	-3	0	0.467825328	84
10	6895	271	257	0	-3	0	-3	-2	2	-2	2	0	0	2	-2	-3	-3	0.638665283	173
14	1197	101	102	2	-3	2	-3	2	2	0	2	2	-2	-2	-2	-3	-3	0.594868474	60
16	5710	447	425	0	-3	0	-3	-2	2	0	0	0	0	0	0	-3	-3	0.61456222	275
19	6368	158	131	-3	-3	0	-3	0	-3	-2	-3	0	-2	0	-2	-3	0	0.687961431	109
20	10210	163	132	-3	-3	0	-3	-2	-2	-2	-3	0	-2	0	0	-3	-3	0.683432382	111
23	4620	248	211	-3	-3	0	-3	0	-3	0	-3	0	-2	0	-2	-3	0	0.688169211	171
26	1259	164	118	1	-3	-3	-3	-3	-1	-3	-3	-3	-2	-3	-3	-3	-3	0.32912973	54
27	11493	449	422	0	-3	-2	-3	-1	1	-2	-1	0	0	-2	0	0	0	0.87003834	391
28	1636	127	83	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	0.424301333	54
30	5700	339	281	-1	-3	2	-3	-3	-3	-1	-3	-1	-2	-2	1	-3	-2	0.700331336	237
33	260	75	56	-3	-3	-1	-1	-3	-3	-3	-3	-3	-1	-1	-3	-3	-2	0.519730461	39
36	1630	264	209	2	-3	-3	-3	-3	2	-3	-3	-3	-3	1	-3	-3	-3	0.264224902	70
38	820	201	139	-2	-3	-2	-3	-3	-2	-3	-3	-3	-3	-2	-3	-3	-3	0.400306437	80
39	4830	220	205	2	-3	1	-1	-3	-2	1	-3	2	-2	-1	-1	-3	-1	0.806960084	178
44	19724	569	467	-3	-3	-3	-3	-3	-1	-3	-1	1	-3	-3	-1	-3	2	0.912432267	519
46	5670	355	320	-1	-3	-3	-3	-1	2	0	1	-1	0	-3	-1	-3	-1	0.545818291	194
47	5535	256	225	1	-3	0	-3	-3	-1	1	-3	0	-3	-3	-1	-3	2	0.668990107	171
49	4604	197	142	-3	-3	-3	-3	-3	-3	-3	-3	-1	-3	-3	-3	-3	2	0.67156519	132
52	3780	108	78	-3	-3	-3	-3	-3	-3	-3	-3	-1	-1	-3	-1	-3	-2	0.634992577	69
54	5049	163	119	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	0.424301333	78
55	2362	140	91	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	0.424301333	59
61	4442	142	112	-3	-3	-1	-3	-3	-3	-3	-3	-1	2	-3	-3	-3	2	0.855750164	122
62	4792	149	106	-3	-3	-3	-3	-3	-3	-3	-3	-1	-3	-1	-2	-3	-2	0.575195953	86
67	3551	120	95	-3	-3	0	-3	-3	-3	-3	-3	0	-3	0	-3	-3	2	0.839198704	101
70	67594	1039	1101	2	-3	2	-3	2	-2	2	0	2	-3	1	0	-3	2	0.799430283	831
71	10759	328	272	-1	-3	2	-3	-3	-3	-3	-3	2	-3	-3	1	-3	-3	0.932450768	306
73	18644	632	525	1	0	0	-3	-3	-3	-3	-3	-1	-1	-2	-2	-3	-1	0.588358051	372
76	11367	459	450	-1	-3	-1	-3	-3	2	2	0	1	-3	-3	-1	-3	2	0.823409733	378
77	6359	220	172	-3	-3	-1	-3	-3	-3	-3	-3	2	-3	-3	-1	-3	-2	0.917379298	202
80	25609	1003	802	-3	-3	-1	-3	-1	0	-3	-1	-1	-3	-3	-3	-3	-1	0.551221848	553
81	12339	599	431	-3	-3	-2	-3	-3	-2	-3	-3	-1	-3	-1	-3	-3	-2	0.558402275	334
94	34317	798	662	0	-3	2	-3	2	2	2	1	1	-3	-2	0	-3	0	0.646838822	516
101	7344	297	287	-1	-3	0	-3	-3	-3	0	-3	1	-2	0	0	-3	0	0.674735161	260
112	1053	60	44	-1	-2	-3	-3	-2	-2	-2	-2	-3	-3	-3	-3	-2	-3	0.409690257	25
114	5468	134	107	-3	-3	2	-3	-3	-3	-3	-3	-1	-3	0	2	-3	-3	0.732683175	98
117	5805	329	234	-3	-3	-3	-3	-3	-3	-3	-3	-1	-3	-3	-1	-3	-3	0.56629654	187
126	2430	177	135	2	-1	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	1	-3	0.523784867	93
133	4172	208	200	0	-3	1	0	1	1	0	-1	-1	-3	0	0	-3	-2	0.463036573	95
136	2800	283	280	2	-2	0	0	-2	2	1	0	0	-3	-3	-1	-3	1	0.608614256	172
141	14973	516	511	-1	-3	0	0	0	-1	0	-1	2	-2	-1	1	-3	1	0.939671285	485
145	3192	101	109	2	-3	0	0	-1	2	1	1	-2	-1	1	1	0	0	0.715966027	72
147	5775	373	380	2	-3	0	0	-1	2	0	0	-3	-1	-1	1	-1	0	0.541578685	202
150	710	172	158	2	-3	-1	-1	-1	2	1	-1	-3	-2	-2	-1	-3	-2	0.328700221	57
151	14987	464	390	1	-3	-2	-2	-1	-3	0	-3	0	-3	0	-1	-3	-3	0.464871746	225
153	2352	794	699	2	-3	-2	-3	-2	2	-1	0	0	-3	-3	0	-3	-3	0.469464464	373
160	9135	443	439	2	-3	-1	-1	0	2	1	1	-1	-3	-1	0	-3	-1	0.449425791	199
161	4620	506	471	2	-3	-1	-1	-2	2	0	1	-3	-3	-3	0	-3	0	0.415984445	210
163	5922	411	366	2	-1	-1	-3	-3	2	0	1	-2	-3	-3	-1	-3	-3	0.415537874	171
168	7000	226	183	-3	-3	0	-1	-1	-3	0	-3	-1	-3	-1	-1	-3	-3	0.577196495	130
169	6876	229	220	1	-2	2	-1	0	2	0	1	-3	-3	-1	-1	-3	-3	0.371560066	85
172	2275	182	142	0	-3	-3	-3	-3	2	-3	-1	-3	-3	-3	-2	-3	-1	0.365321467	66
174	9100	596	542	1	-3	-1	-3	-2	2	-2	1	1	-2	-2	-2	-3	-1	0.646694159	387
176	595	293	208	-3	-3	-3	-3	-3	2	-3	-3	-3	-2	-3	-3	-3	-3	0.357557061	105
182	2969	183	152	-1	-3	-2	-2	-3	2	-2	1	-3	0	-3	-2	-3	-3	0.450763371	82
185	12824	318	369	2	1	-1	2	0	2	1	0	-1	0	0	1	1	1	0.676742751	279
193	8512	151	156	2	-3	0	-1	0	2	-1	1	-2	-3	0	1	0	0	0.626570948	95

Project No.	Effort(mh)	UFP	AFP_{old}	GSC(adj)1	GSC(adj)2	GSC(adj)3	GSC(adj)4	GSC(adj)5	GSC(adj)6	GSC(adj)7	GSC(adj)8	GSC(adj)9	GSC(adj)10	GSC(adj)11	GSC(adj)12	GSC(adj)13	GSC(adj)14	Overall Multiplier	AFP_{new}
198	1876	135	108	-1	-3	-2	-3	-3	2	-3	-2	-1	-3	-3	-2	-3	0	0.505501193	68
201	2723	165	130	-1	-3	-2	-3	-3	2	-3	-2	-1	-3	-3	-2	-3	-1	0.483884616	80
205	2583	146	121	0	-3	-1	-3	-3	1	-1	-2	0	-3	-3	-2	-3	-1	0.560652768	82
212	3521	395	431	0	0	1	1	1	2	0	1	0	1	0	-1	-1	-3	0.754606706	298
217	2632	110	103	0	0	-3	-3	2	-1	-1	0	0	0	-1	-3	-3	0	0.52549991	58
218	2303	104	90	1	0	1	-3	-3	2	-3	-2	-2	-3	-1	-3	-2	-2	0.452227389	47
225	3164	305	299	0	-3	0	1	0	2	0	0	-1	-3	-2	-1	-3	1	0.53796092	164
228	3542	301	307	2	-3	1	0	-1	2	-1	0	-2	0	-3	-1	0	1	0.734970313	221
230	7252	281	259	-1	-3	0	1	0	-2	0	0	-1	-3	-3	-1	-3	1	0.669124246	188
231	3927	202	186	-1	-3	1	0	0	0	0	0	-3	-2	-3	-1	-3	0	0.492090321	99
232	10577	371	386	1	-3	-1	1	-1	2	1	1	0	-2	0	0	-3	1	0.65069186	241
239	1267	80	75	2	0	0	-3	-2	2	-1	0	0	-2	0	-3	-3	-3	0.48127684	39
240	11361	514	524	2	0	0	-3	0	1	0	1	-2	-1	1	-1	-3	0	0.456900459	235
247	1674	67	59	-3	-3	-3	-3	-3	-2	-3	-3	-1	-3	-3	-3	-3	-3	0.518082826	45
256	8248	149	112	-2	-3	-3	-3	-3	-2	-1	-2	-1	-3	-3	-1	-3	-2	0.568142838	85
257	1418	67	61	-3	-3	-3	-3	-3	-3	-3	-3	-1	2	-3	-1	-3	-3	0.672369767	45
259	6412	201	141	-3	-3	-3	-3	-3	-3	-3	-3	-1	-3	-3	-1	-3	-3	0.56829654	114
262	16092	1257	1006	-3	-3	2	-3	-1	-3	-3	-3	1	-3	-3	-1	-1	-3	0.960173672	1232
264	5710	221	192	-1	-3	0	-3	-3	-3	0	-3	1	-2	0	-3	-3	0	0.809642978	179
267	1161	63	42	-3	-3	-3	-3	-3	-3	-3	-3	-1	-3	-3	-3	-3	-3	0.539742138	34
269	1067	169	132	-3	-3	-3	-3	-3	-3	-3	-3	0	-2	-2	-3	-3	-3	0.627734967	119
270	3267	173	140	-3	-3	0	0	-3	-3	-3	-3	-3	-2	0	-2	-3	0	0.589225714	102
283	7304	223	178	-3	2	-3	-3	-3	-1	1	-1	-1	-3	-3	-3	-3	-3	0.550716978	123
286	1601	190	124	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	0.424301333	81
288	2124	417	271	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	0.424301333	177
289	8165	1206	784	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	0.424301333	512
291	4856	669	435	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	0.424301333	284
294	1596	302	196	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	0.424301333	128
295	3084	373	242	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	0.424301333	158
296	1744	322	209	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	0.424301333	137
298	5806	196	127	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	0.424301333	83

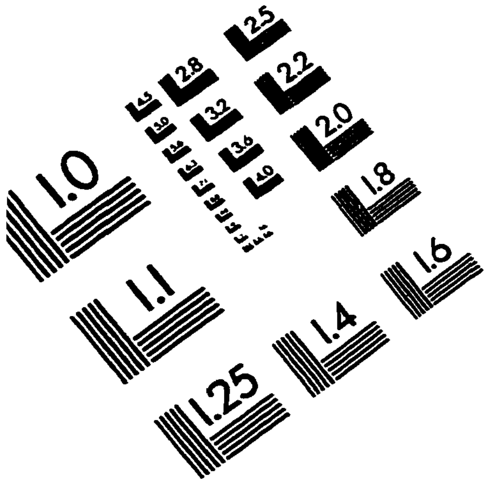
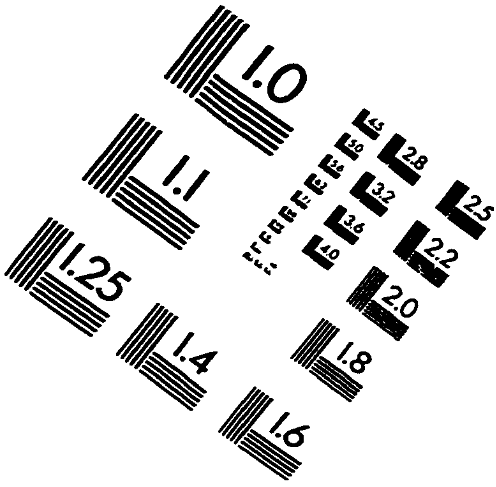
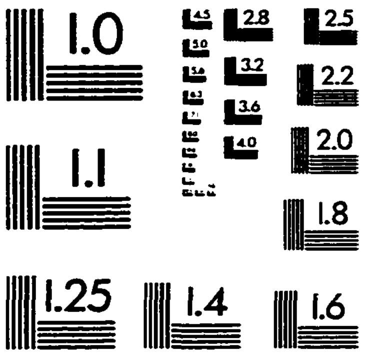
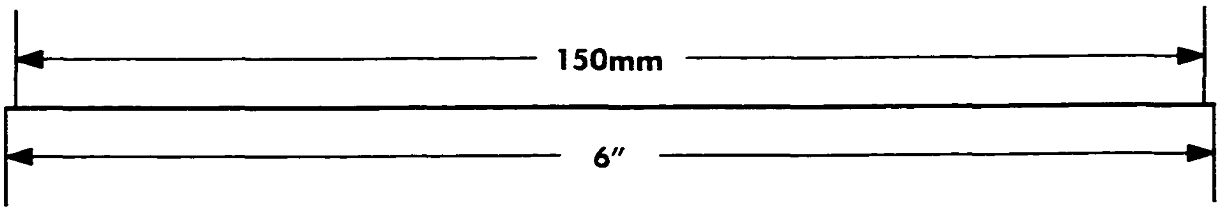
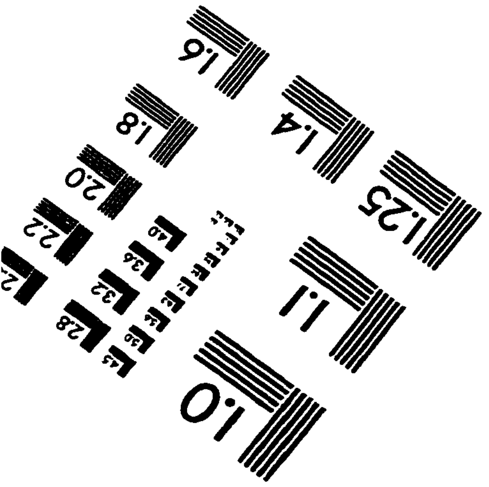
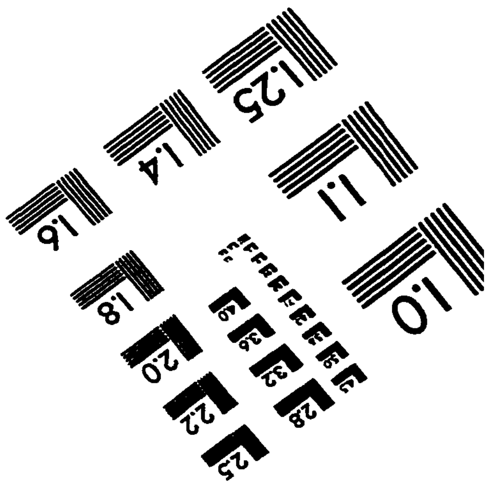
VITA

William Alexander Eldred is a Principal Engineer with PRC Inc., an information management and technical services firm, at the company's Virginia Beach, Virginia, location. He is a former Engineering Duty Officer in the U. S. Navy. Since completing a twenty year Navy career in 1982, he has pursued a career as a technical manager in the private sector. He is a licensed Professional Engineer in the Commonwealth of Virginia.

His education includes a Bachelor of Science degree from the U. S. Naval Academy (June, 1962), a Master of Science degree in Management from the Sloan School of Management at the Massachusetts Institute of Technology (June, 1972), the degree of Ocean Engineer from the Massachusetts Institute of Technology (June, 1972), and the Doctor of Philosophy degree in Engineering Management from Old Dominion University in Norfolk, Virginia (December, 1998). He was elected to membership in Tau Beta Pi, Massachusetts Beta Chapter, in 1972.

William Alexander Eldred was born on May 1, 1939, in Glasgow, Kentucky. He currently resides in Virginia Beach, Virginia, with his wife Judith. They have three adult children, John and Marshall, both of whom live in Richmond, Virginia, and Alina, who is a graduate student at The American University in Washington, DC.

IMAGE EVALUATION TEST TARGET (QA-3)

APPLIED IMAGE, Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989