Management of Computing

Robert W. Zmud Editor

An Empirical Validation of Software Cost Estimation Models

CHRIS F. KEMERER

ABSTRACT: Practitioners have expressed concern over their inability to accurately estimate costs associated with software development. This concern has become even more pressing as costs associated with development continue to increase. As a result, considerable research attention is now directed at gaining a better understanding of the software-development process as well as constructing and evaluating software cost estimating tools. This paper evaluates four of the most popular algorithmic models used to estimate software costs (SLIM, COCOMO, *Function Points, and* ESTIMACS). *Data on 15 large completed business data-processing projects were collected and used to test the accuracy of the models'* ex post *effort estimation. One important result was that Albrecht's Function Points effort estimation model was validated by the independent data provided in this study* [3]. *The models not developed in business data-processing environments showed significant need for calibration. As models of the software-development process, all of the models tested failed to sufficiently reflect the underlying factors affecting productivity. Further research will be required to develop understanding in this area.*

1. INTRODUCTION

Practitioners have expressed concern over their inability to accurately estimate costs associated with software development. This concern has become even more pressing as costs associated with development continue to increase. As a result, considerable research attention is now directed at gaining a better understanding of the software-development process as well as constructing and evaluating software cost estimating tools.

© 1987 ACM 0001-0782/87/0500-0416 75¢

This paper presents an empirical validation of four algorithmic models from the literature (SLIM, COCOMO, Function Points, and ESTIMACS) that are proposed as general software-development cost estimators. This research has built a database of completed software-development projects and focuses on comparing the actual costs with the *ex post* estimates obtained from the four models. Of particular interest is the comparison of the results of the four models' use of different measures (metrics) of the outputs underlying their productivity calculations.

Three research questions of interest to practitioners are addressed in this paper:

(1) Are these software cost estimating models truly generalizable to environments other than that in which they were developed? If not, can they be easily calibrated to a typical business data-processing environment?
(2) Are models that do not use source lines of code (SLOC) as an input as accurate as those that do? If so, then this could eliminate the need to attempt to estimate lines of code early in the project.
(3) Are the models available in the open literature as accurate as proprietary models? If so, then this could allow practitioners to avoid the costs of purchasing the proprietary models.

Section 2 discusses the selection of the four models, followed by brief descriptions of the models. Section 3 describes the environment in which the data originated and compares this data set with that of other researchers. Section 4 describes the data-gathering and data-analysis methods, which leads directly into Section 5, "Results." Finally, Section 6

offers some summary conclusions and outlines extensions of this work.

2. BACKGROUND TO THE MODELS

2.1 Model Selection

A review of the literature revealed that the most interesting difference between estimation models is between models that use SLOC as the primary input versus models that do not. SLOC was selected early as a metric by researchers, no doubt due to its quantifiability and seeming objectivity. Since then an entire subarea of research has developed to determine the best method of counting SLOC [10, 12]. In response to this, and to practitioners' complaints about the difficulties in estimating SLOC before a project was well under way, new models have been developed that do not use SLOC as the primary input [2, 20].

The next step was to choose models that were well regarded in the current literature, and to winnow this selection down to a manageable number since each model's idiosyncrasies required the collection of different data. The approach to determining which models were well regarded was twofold: First, Barry Boehm (developer of the COCOMO model, a model eventually selected for this study) has written a widely cited book entitled *Software Engineering Economics* [5] in which he provides an analysis of eight important models. This list was used to generate candidates. The second step was a review of articles in the most recent issues of the *Journal of Parametrics*, a publication of the International Society of Parametric Analysts that devotes a large number of its pages to software estimation articles. This sampling validated whether the models cited in the Boehm study were well represented by their inclusion in other studies.

Boehm examines eight models (including COCOMO) in his evaluation. They are (1) SDC, (2) Wolverton, (3) SLIM, (4) Doty, (5) PRICE, (6) IBM-FSD, (7) Boeing, and (8) COCOMO [5, 16, 25, 27]. After examining this list of candidates, a review was undertaken of the most recent issues of the *Journal of Parametrics*, to determine the popularity of these models as demonstrated by their inclusion in other research. The results are presented in Table I with an "X" signifying that a particular model was discussed or compared in that article [7, 9, 14, 15, 21].

From this list of candidates, COCOMO and SLIM seem to be the most widely reviewed. The PRICE model is also popular, but was developed primarily for use on aerospace applications and was therefore deemed unsuitable for the business applications that would comprise the database. Therefore, only COCOMO and SLIM were selected for this research.

TABLE I. *Journal of Parametrics* Articles

	COCOMO	SLIM	PRICE
Callisen (12/84)	X		
Ferens (12/84)	X	X	X
Masters (3/85)	X	X	X
Pinsky (12/84)	X	X	
Rubin (6/85)	X	X	

All of the above analysis centered on SLOC models of one sort or another. This is to be expected since software effort estimation research began with these models and is a relatively young field. No similar comparisons for non-SLOC models were found, and in fact, only two models were discovered during the period when this research was being conducted. These two are the Function Points method, developed by Allan Albrecht at IBM in 1979 [2, 3], and the ESTIMACS model, developed by Howard Rubin of Hunter College and marketed by Management and Computer Services during the period when these data were being collected [20, 22]. Both of these models were selected, bringing the total number of models compared to four.

These four exhibit a certain symmetry, perhaps best illustrated in Table II. Therefore, some interesting comparisons can be made between not only measurement methods, but between models that appear in the open literature and those that do not.

TABLE II. Model Categories

	Proprietary	Nonproprietary
SLOC	SLIM	COCOMO
Non-SLOC	ESTIMACS	Function Points

2.2 Models

Due to space limitations, the following are necessarily brief descriptions of the four models used in this paper. The interested reader is referred to the references provided for more detailed explanations.

2.2.1 *SLIM.* The SLIM estimating method was developed in the late 1970s by Larry Putnam of Quantitative Software Management [16, 17]. SLIM depends on an SLOC estimate for the project's general size, then modifies this through the use of the Rayleigh curve model to produce its effort estimates. The user can influence the shape of the curve through two key parameters: the initial slope of the curve (the "manpower buildup index" or MBI) and a productivity factor (the "technology constant" or PF). Since these are dimensionless numbers, the SLIM

user has two options for choosing their values: The user can calibrate the model by inputting data from completed projects, or he or she can answer a series of 22 questions, from which SLIM will provide a recommended PF and MBI. The second method was chosen for this research due to the absence of a previously collected calibration database and the feeling that this would more accurately reflect the average user's experience.

2.2.2 *COCOMO.* The COnstructive COst MOdel (COCOMO) was developed by Barry Boehm of TRW and published in 1981 [5]. Based on his analysis of 63 software-development projects, Boehm developed an easy-to-understand model that predicts the effort and duration of a project, based on inputs relating to the size of the resulting systems and a number of "cost drivers" that Boehm believes affect productivity.

A simplified version of the essential COCOMO effort equation for the Basic Model (the Intermediate and Detailed Models are discussed later) is of the form

$$MM = C\,(KDSI)^k,$$

where

MM = number of man-months[1] (= 152 working hours),
C = a constant,
KDSI = thousands of "delivered source instructions" (DSI), and
k = a constant.

Boehm defines DSI as program instructions created by project personnel that are delivered as part of the final product. They exclude comments and unmodified utility software, and include job control language, format statements, and data declarations.

In Boehm's development of COCOMO, he found that the Basic Model predicted effort within a factor of 1.3 only 29 percent of the time and within a factor of 2 only 60 percent of the time for his 63-project database. In an effort to improve the model's accuracy, he refined the equation to include the effects of 15 "cost drivers," which are attributes of the end product, the computer used, the personnel staffing, and the project environment. He believes that these 15 factors affect the project's productivity and calls this version the Intermediate Model.

The COCOMO Detailed Model is very similar to the Intermediate Model except that the project is divided into four phases: Product Design, Detailed Design, Coding/Unit Test, and Integration/Test. The 15 cost drivers are estimated and applied to each phase separately, rather than to the project as a whole.

2.2.3 *Function Points.* The Function Points measurement method was developed by Allan Albrecht at IBM and first published in 1979 [2, 3]. Albrecht was interested in the general problem of productivity measurement in systems development and created the Function Points method as an alternative to estimating SLOC. Albrecht's Function Points are at a more macro level than SLOC, capturing things like the number of input transaction types and the number of unique reports. He believes Function Points offer several significant advantages over SLOC counts: First, it is possible to estimate them early in the life cycle, about the time of the requirements definition document. This can be an important advantage for anyone trying to estimate the level of effort to be required on a software-development project. Second, they can be estimated by a relatively nontechnical project member. And, finally, they avoid the effects of language and other implementation differences.

There are two steps involved in counting Function Points: (1) counting the user functions, and (2) adjusting for processing complexity. There are currently five user function categories:[2] external input types, external output types, logical internal file types, external interface file types, and external inquiry types. Albrecht recognized that the effort required to provide a given level of functionality can vary depending on the environment. For example, input transactions are harder to program if a lot of emphasis is placed on system throughput or end-user convenience. In response to this, Albrecht has a list of 14 processing complexity characteristics that are to be rated on a scale from 0 (no influence) to 5 (strong influence). The next step is to sum all the processing complexity points assigned. This number is then multiplied by 0.01 and added to 0.65 to obtain a weighting, as follows:

$$PCA = 0.65 + 0.01 \sum_{i=1}^{14} c_i,$$

where PCA = processing complexity adjustment ($0.65 \le PCA \le 1.35$), and c_i = complexity factors ($0 \le c_i \le 5$).

This factor is then used in the final equation:

$$FP = FC(PCA),$$

where FP = Function Points, and FC = previously computed Function Counts.

The end result is that the Function Points can vary ±35 percent from the original Function Counts.

[1] "Man-months" is used in this paper in its vernacular sense of referring to all the hours worked on a project, whether by men or women.

[2] Albrecht's 1979 paper has only four, omitting External Interface File Types [2].

Once the Function Points have been computed, they can be used to compare the proposed project with past projects in terms of its size.

2.2.4 *ESTIMACS.* The ESTIMACS model was developed by Howard Rubin of Hunter College as an outgrowth of a consulting assignment to Equitable Life. It is a proprietary system and, at the time the data were collected, was marketed by Management and Computer Services (MACS). Since it is a proprietary model, details, such as the equations used, are not available. The model does not require SLOC as an input, relying instead on "Function-Point-like" measures. The research in this paper is based on Rubin's paper from the 1983 IEEE conference on software development tools and the documentation provided by MACS [20, 22]. The 25 ESTIMACS input questions are described in these documents.

3. DATA

3.1 Data Source

The source for the project data for this research was a national computer consulting and services firm specializing in the design and development of data-processing software (hereafter referred to as the ABC consulting firm). Use of this consulting firm's data conveyed several advantages to this research.

First, because clients are charged by the hours spent on their projects, the consulting firm must maintain strict, auditable timekeeping records. Since a key element of this research is the comparison of estimated effort to actual effort, it was important that the measurement of actual effort be as accurate as possible.

Second, the projects in the database reflect a variety of data-processing applications, which should broaden the level of interest in the results, as opposed to, say, a database composed of all insurance or all defense contractor applications. However, within this framework there is a high level of consistency in both the quality of the staff and the methodologies employed, since all the projects were done by the same firm within the time span of a few years.

Third, as professional software developers, ABC is highly interested in software productivity. Since it is their primary business, as opposed to being relegated to a staff function as it might be in another industry, the ABC managers were highly motivated to provide good quality data and cooperate with the research.

3.2 Project Database

Projects selected possessed two attributes: First, they were medium to large in size, an explicit requirement of the models chosen [5, 19]. The average project size in this study is just under 200 KSLOC (SLOC stated in thousands), and the smallest project is 39 KSLOC, which was partly written in a fourth-generation language. Second, the project manager must have been available during the summer of 1985 to complete the data-collection form. (See Section 4 for more details.) This limited the database to fairly recent projects, which had the additional benefit of ensuring consistency in project methodology. The oldest project in the database was started in August 1981, and two-thirds of the projects started in 1983 or later.

Based on the above criteria, 15 projects qualified for inclusion in the study. The researcher or practitioner interested in large business applications is likely to find this data set to be a useful contribution for two reasons: First, in terms of sheer number of projects, it compares favorably with a number of other published studies [4, 7, 9, 11]. Second, and more important than the number of projects alone, is the content of the database. This database contains large business applications, 12 of which were written entirely in Cobol, the most widely used business data-processing language. By contrast, Boehm's 63-project database contains only 7 business applications, of which only 4 were written in Cobol. Albrecht's 1983 database contains 18 Cobol projects, but with an average size of only 66 KSLOC. The average size of Cobol applications in the ABC database is 221 KSLOC.

4. METHODOLOGY

4.1 Data-Collection Forms

The primary data-collection tools were three data-collection forms developed for this project. One form was designed to capture background information about the project (e.g., hardware manufacturer and model), as well as the ABC project number and the start and end dates.

After an analysis of each of the model's requirements, two additional consolidated metrics data-collection forms were devised. Although each of the models has its own set of data requirements, there is a fair amount of overlap in terms of the areas addressed as they relate to productivity. The questions were organized on the forms into broad categories, such as personnel attributes, rather than by models in order to help the respondent ensure consistency in his or her answers for all of the models. However, the questions were printed on the form verbatim from the original source documents whenever possible to minimize any effects of orginality in the wording by the researcher.[3] The net result was a set of two data-collection forms, one "quantitative" (e.g.,

[3] The original source documents were, for SLIM, [19]; COCOMO, [5]; Function Points, [3]; and ESTIMACS, [22].

SLOC, database size) and one "qualitative" (e.g., complexity ratings). The intent of this procedure was to retain the authenticity of the questions as asked in the original source documents, while providing some overall structure for the respondents.

4.2 Data-Collection Procedure

For each of the projects, there was an in-person meeting with the project manager who would be filling out the forms. The one exception to this procedure was a West Coast project manager who was already familiar with the metrics used. Data collection for that particular project was done via the mail and the telephone.

There were two main purposes to this labor-intensive approach. The first was to discuss each of the questions to ensure that it was understood and that each of the managers would answer consistently. The second purpose was to impress upon the managers the importance of their participation in this work. Given the amount of data requested (greater than 100 questions per project), this effort was believed necessary. The result was a large increase in the response rate compared to questionnaires distributed by the normal channels at ABC.

4.3 Data-Analysis Procedures

Once the forms were returned, they were checked for consistency. This was possible due to the redundancy of many of the questions from the different models. With the exception of the "Function-Point-like" questions of the ESTIMACS model, no questions were combined or consolidated.[4] This meant that it was possible to ensure that an answer about, say, project personnel skills for the COCOMO model was consistent with the answer for the similar question for the SLIM model. Next, the project identification information from the "Project Actuals" form was used to research the ABC accounting records to determine the level of effort expended. Finally, the metrics data were input to each of the four models. For SLIM, COCOMO, and Function Points, this was done by the researcher. MACS staff offered to run a limited number of projects (without having access to the actual results) and to provide the estimates by telephone.

4.4 Error Analysis

The focus of this paper is on the degree to which the model's estimated effort (MM_{est}) matches the actual effort (MM_{act}). If the models were perfect, then for every project $MM_{est} = MM_{act}$. Clearly, this will rarely, if ever, be the case.

[4] See the ESTIMACS results section (Section 5.5) for discussion of this issue.

There are several possible methods for evaluating the man-month estimates. A simple analysis approach would be to look at the difference between MM_{est} and MM_{act}. The problem with this *absolute error* approach is that the importance of the size of the error varies with project size. For example, on a 10 man-month project, an absolute error of 9 man-months would be likely to cause serious project disruption in terms of staffing, whereas the same error on a 1000 man-month project would be much less of a problem.

In light of this, Boehm [5] and others have recommended a *percentage error* test, as follows:

$$\text{Percentage Error} = \frac{MM_{est} - MM_{act}}{MM_{act}}$$

This test eliminates the problem caused by project size and better reflects the impact of any error.

However, the analysis in this paper concentrates on the models' average performance over the entire set of projects. Errors can be of two types: underestimates, where $MM_{est} < MM_{act}$; and overestimates, where $MM_{est} > MM_{act}$. Both of these errors can have serious impacts on projects. Large underestimates will cause the project to be understaffed, and as the deadline approaches, project management will be tempted to add new staff members. This results in a phenomenon known as Brooks's law: "Adding manpower to a late software project makes it later" [6]. Otherwise productive staff are assigned to teaching the new team members, and with this, cost and schedule goals slip even further. Overestimates can also be costly in that staff members, noting the project slack, become less productive (Parkinson's law: "Work expands to fill the time available for its completion") or add so-called "gold plating," defined as additional systems features that are not required by the user [5].

In light of the seriousness of both types of errors, overestimates and underestimates, Conte et al. [8] have suggested a *magnitude of relative error*, or *MRE* test, as follows:

$$\text{MRE} = \left| \frac{MM_{est} - MM_{act}}{MM_{act}} \right|$$

By means of this test, the two types of errors do not cancel each other out when an average of multiple errors is taken, and therefore as the test used in this analysis.

Still another issue in interpreting the errors concerns bias. A model, because it was developed in an environment, say, less productive than ABC's, may generate errors that are biased toward overestimation. However, this model may be able to be recalibrated to approximate the ABC environment. What

is important is that the estimates correlate with the actual results; that is, bigger projects generate bigger estimates than smaller projects.

Albrecht and others [3, 4, 8] have proposed linear regression as a means of measuring this correlation. Since Albrecht's method does not produce a man-month estimate directly, an alternative was needed to validate the Function Point method. His proposal was to perform a simple linear regression with man-months as the dependent variable and Function Points as the independent variable. He also performs regressions with SLOC as the dependent variable to show how Function Points could be used to generate an SLOC estimate for other models or methods that require it. Regression was used in this research for all of the models by using actual man-months as the dependent variable and the man-months estimated by each model as the independent variable. The advantage of this method is that it can show whether a model's estimates correlate well with experience even when the MRE test does not. A "perfect" score on the MRE test would be an error percentage of 0, whereas a "perfect" score on Albrecht's test would be an R^2 of 1.00.[5]

In summary, two tests are used to evaluate the models (as recommended by Theabaut [24]): the MRE test and regression. These tests have the advantage of not only being generally accepted in the literature, but of also having been proposed by some of the models' own developers.

[5] In this analysis the more conservative $\bar{R}^2$, R^2 adjusted for degrees of freedom, was used, which results in slighter lower values than R^2.

5. RESULTS

5.1 Project Data

Table III shows the background data on the projects. The "Months" figures are project durations in calendar months. The man-months ("MM") data refer to the total number of actual hours expended by exempt staff members (i.e., not including secretarial labor) on the project through implementation, divided by 152. This effort measure was the only output of the models that was evaluated, although several of them offer other features, such as project planning aids. The "KSLOC" figures follow Boehm's definition stated earlier (i.e., comments are not counted) with two exceptions.

One project (number 8) had reused code, and this was accounted for in the KSLOC data by counting a modified line as equivalent to a new line. In the KDSI data (used for COCOMO), Boehm's conversion formula for modified software was used [5]. This resulted in a KDSI count of 167 as opposed to the 200 KSLOC count shown above for project number 8.

The other exception is that Cobol nonprocedural statements are not specially weighted. Boehm found this weighting necessary as only 4 of his 63 projects were Cobol, and as he states, some "pragmatic" procedure was required to make those projects fit his formula.[6] Since the project data in this research are overwhelmingly Cobol, no such pragmatic weighting was deemed necessary.

[6] See [5, p. 479]. The unavailability of these data may make the COCOMO estimates slightly higher than they would be elsewise.

TABLE III. Project Background

Project number	Software	Hardware	Months	MM	KSLOC	SLOC/MM
1	Cobol	IBM 308X	17	287.00	253.60	884
2	Cobol	IBM 43XX	7	82.50	40.50	491
3	Cobol	DEC VAX	15	1,107.31	450.00	406
4	Cobol	IBM 308X	18	86.90	214.40	2,467
5	Cobol	IBM 43XX	13	336.30	449.90	1,338
6	Cobol	DEC 20	5	84.00	50.00	595
7	Bliss	DEC 20	5	23.20	43.00	1,853
8	Cobol	IBM 43XX	11	130.30	200.00	1,535
9	Cobol	IBM 308X	14	116.00	289.00	2,491
10	Cobol, Natural	IBM 308X	5	72.00	39.00	542
11	Cobol	IBM 308X	13	258.70	254.20	983
12	Cobol	IBM 43XX, 308X	31	230.70	128.60	557
13	Cobol	HP 3000, 68	20	157.00	161.40	1,028
14	Cobol	IBM 308X	26	246.90	164.80	667
15	Natural	IBM 308X	14	69.90	60.20	861
Mean			14.3	219.25	186.57	1,113
Standard deviation			7.5	263.06	136.82	689
Cobol mean					221.37	1,120
Cobol standard deviation					131.14	720

TABLE IV. SLIM Data

Project number	Actual MM	SLIM MM estimate	SLIM error (%)	PF	MBI	Calibrated PF	Calibrated MBI
1	287.0	3,857.8	1,244.18	12	5	18	3
2	82.5	100.1	21.33	15	4	16	5
3	1,107.3	11,982.0	982.08	16	4	19	5
4	86.9	2,017.2	2,221.29	13	4	18	1
5	336.3	3,382.0	905.65	15	5	21	4
6	84.0	262.5	212.50	13	4	19	6
7	23.2	106.3	358.19	16	5	20	5
8	130.3	1,224.6	839.83	14	4	20	3
9	116.0	1,454.1	1,153.53	15	4	20	2
10	72.0	235.7	227.36	12	4	18	6
11	258.7	1,623.0	527.37	15	5	19	3
12	230.7	513.3	122.50	15	4	12	1
13	157.0	3,119.8	1,887.13	11	5	16	1
14	246.9	380.3	54.03	17	4	14	1
15	69.9	643.8	821.03	11	4	15	1
Mean (absolute values)	219.25	2,060.17	771.87	14.0	4.3	17.7	3.1
Standard deviation	263.06	3,014.77	661.33	1.9	0.5	2.6	1.9

5.2 SLIM Results

SLIM was run using the default parameters provided by the model when the user answers the SLIM set of 22 questions. Table IV shows the actual man-months, the SLIM man-month estimate,[7] and the error percentage. Also shown are the SLIM default PF, the default MBI, the calibrated PF, and the calibrated MBI. (The calibrated factors were obtained after the fact by using the SLIM "calibrate" function for use in later analyses.)

SLIM does not do well via the MRE test. The average percentage error is 772 percent, with the smallest error being 21 percent. In addition, the errors are all biased; effort is overestimated in all 15 cases. A possible explanation for this is that SLIM was originally developed using data from defense-related projects, including real-time systems. Productivity for these is typically less than that in the business data-processing systems developed by ABC [18, 23].

One possibility is that SLIM accurately reflects the project conditions, but is calibrated too high for the ABC environment. To evaluate this Albrecht's test was applied by means of a regression run with the SLIM estimate as the independent variable and the actual man-months as the dependent variable. The results were that the SLIM estimates correlated well with the actuals, generating an $\bar{R}^2$ of 87.8 percent, with a coefficient t statistic of 10.11.

Actual man-months = 49.9 + 0.082 (SLIM) (10.11) — $\bar{R}^2$ 87.8%

[7] The SLIM man-month is 168 man-hours. All SLIM man-month estimates were converted to 152 man-hour man-months to aid comparison.

The main input for SLIM is KSLOC, whose regression with actual man-months produced an $\bar{R}^2$ of only 48.5 percent, so the SLIM model seems to be adding information.

5.3 COCOMO Results

The results for the three COCOMO versions (Basic, Intermediate, and Detailed Models) appear in Table V. In the table "BAS" refers to the Basic Model, "INT" to the Intermediate Model, and "DET" to the Detailed Model. The "error %" columns are calculated by dividing the difference between the estimate and the actual man-months by the actual man-months.

All three COCOMO models did poorly according to the MRE percentage error test. The average error for all versions of the model was 601 percent, with the lowest single error being 83 percent. As was the case with SLIM, the estimates are biased; effort is overestimated in all 45 cases. Again, this may be due to COCOMO's development on TRW's data. The average KSLOC/MM figure for Boehm's data is much lower than that for the ABC data, probably reflecting the composition of the type of systems developed by TRW and ABC. Again, it is possible that COCOMO is accurately reflecting the project conditions, but is calibrated too high for the ABC environment. In his book Boehm acknowledges the probable need for calibration of his model in a new environment [5].

To evaluate this Albrecht's regression test was run. The results were as follows:

Actual man-months = 27.7 + 0.156 (COCOMO-Basic) (5.54) — $\bar{R}^2$ 68.0%

Actual man-months
= 62.1 + 0.123 (COCOMO–Intermediate) 59.9%
(4.68)

Actual man-months
= 66.8 + 0.118 (COCOMO–Detailed) 52.5%
(4.06)

Paradoxically, the more advanced versions, COCOMO–Intermediate and COCOMO–Detailed, do not do as well as COCOMO–Basic in this instance. This implies that the cost drivers of the latter two models are not adding any additional explanation of the phenomenon. This is consistent with the results of Kitchenham and Taylor, who evaluated COCOMO on some systems programming and real-time systems projects developed by British Telecom and ICL [13].

In addition, a regression was run on the primary input to the COCOMO models, delivered source instructions, or KDSI. The result was an $\bar{R}^2$ of 49.4 percent, which is not as good a fit as any of the COCOMO models. Therefore, COCOMO seems to be adding information.

5.4 Function Points Results

In this section three separate analyses are performed. The first analysis compares the man-month predictions from Function Points to the actual man-months and is similar to the analysis done for the other models. In addition, two of Albrecht's other models are tested, one for predicting man-months from SLOC, and the other for predicting SLOC from Function Points.

5.4.1 *Function Points to Man-Month Results.* As shown in Table VI, the average MRE is 102.74 percent, substantially better than either of the two SLOC-based models. This is probably due to the similarity of applications (business data processing) done by ABC and IBM's DP Services group, the source of Albrecht's data. The regression analysis of these estimates and the actual man-months was the following:

Actual (ABC) man-months $\bar{R}^2$
= −37 + 0.96 (Function Points 55.3%
(4.26) estimated man-months)

The regression analysis result from the ABC data is as follows:

ABC man-months $\bar{R}^2$
= −122 + 0.341 (Function Points) 55.3%
(4.28)

By way of comparison, in Albrecht's validation paper [3] he developed the following estimation equation from a data set of 24 projects:

IBM man-months $\bar{R}^2$
= −88 + 0.355 (Function Points) 86.9%
(12.37)

Albrecht's equation developed using IBM data is strikingly similar to the equation using ABC data, differing primarily by the constant term. The results of an *F*-test comparing the two individual regressions with a single regression on all 39 data points revealed that the null hypothesis that the two models were the same could not be rejected at the 95 percent confidence level [26].

TABLE V. COCOMO Data

Project number	Actual MM	COCOMO BAS	COCOMO-BAS error (%)	COCOMO INT	COCOMO-INT error (%)	COCOMO DET	COCOMO-DET error (%)
1	287.00	1,095.10	281.57	917.56	219.71	932.96	225.07
2	82.50	189.40	129.58	151.66	83.83	151.19	83.26
3	1,107.31	5,497.40	396.46	6,182.65	458.35	5,818.75	425.49
4	86.90	1,222.30	1,306.56	558.98	543.25	566.50	551.90
5	336.30	1,466.00	335.92	1,344.20	299.70	1,316.04	291.33
6	84.00	393.60	368.57	313.36	273.05	312.24	271.71
7	23.20	328.40	1,315.52	234.78	911.98	234.51	910.82
8	130.30	925.30	610.13	1,165.70	794.63	1,206.17	825.69
9	116.00	3,231.00	2,685.34	4,248.73	3,562.70	4,577.62	3,846.22
10	72.00	181.60	152.22	180.29	150.40	181.36	151.89
11	258.70	1,482.20	472.94	1,520.04	487.57	1,575.68	509.08
12	230.70	691.00	199.52	558.12	141.92	584.37	153.30
13	157.00	891.20	467.64	1,073.47	583.74	1,124.36	616.15
14	246.90	511.00	106.97	629.22	154.85	663.84	168.87
15	69.90	295.30	322.46	133.94	91.62	130.72	87.01
Mean (absolute values)	219.25	1,226.72	610.09	1,280.85	583.82	1,291.75	607.85
Standard deviation	263.06	1,412.58	684.55	1,698.43	862.79	1,667.24	932.96

TABLE VI. Function Points-MM Data

Project number	Actual MM	Function Points	Function Counts	Albrecht FP MM estimate	Albrecht MM error (%)
1	287.00	1,217.10	1,010.00	344.30	19.96
2	82.50	507.30	457.00	92.13	11.68
3	1,107.31	2,306.80	2,284.00	731.43	−33.95
4	86.90	788.50	881.00	192.03	120.98
5	336.30	1,337.60	1,583.00	387.11	15.11
6	84.00	421.30	411.00	61.58	−26.69
7	23.20	99.90	97.00	−52.60	−326.73
8	130.30	993.00	998.00	264.68	103.13
9	116.00	1,592.90	1,554.00	477.81	311.90
10	72.00	240.00	250.00	−2.83	−103.93
11	258.70	1,611.00	1,603.00	484.24	87.18
12	230.70	789.00	724.00	192.21	−16.68
13	157.00	690.90	705.00	157.36	0.23
14	246.90	1,347.50	1,375.00	390.63	58.21
15	69.90	1,044.30	976.00	282.91	304.73
Mean (absolute values)	219.25	999.14	993.87	266.87	102.74
Standard deviation	263.06	569.60	577.17	202.36	112.11

A similar regression was performed on the ABC Function Count data, the main input to Function Points.

ABC man-months $\bar{R}^2$
$= -111 + 0.333$ (Function Counts) 53.8%
(4.16)

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.

5.4.2 *Source Lines of Code to Man-Month Results.* Albrecht provides another model, showing the relationship for his data between Cobol KSLOC and man-months.

IBM man-months $\bar{R}^2$
$= -19.2 + 2.5$ (Cobol KSLOC) 73.0%
(6.85)

The results are shown in Table VII.

The average MRE of 167.3 percent is not as good as that generated by Function Points. The equivalent equation for the ABC data is the following:

ABC man-months $\bar{R}^2$
$= -65 + 1.47$ (Cobol KSLOC) 41.9%
(2.99)

The results of an *F*-test are that we can reject the null hypothesis that the IBM and ABC models are the same at the 95 percent confidence level, but not at the 99 percent level.

5.4.3 *Function Points to Source Lines of Code Results.* Another validation that was performed was a check of Albrecht's claim of Function Points as a predictor of KSLOC. Albrecht provides the following model for estimating Cobol SLOC from Function Points:

IBM Cobol KSLOC $\bar{R}^2$
$= -6.5 + 0.12$ (Function Points) 71.2%
(6.55)

The results are shown in Table VIII, and show an average MRE of 38.17 percent, with a large negative bias. The estimated relationship between the actual KSLOC and the estimates is the following:

ABC Cobol KSLOC $\bar{R}^2$
$= 6 + 1.68$ (Function Points 65.6%
(4.69) estimated KSLOC)

In general, these results seem to validate Albrecht's claims that Function Points do correlate

TABLE VII. Cobol KSLOC-MM Data

Project number	Actual MM	Albrecht SLOC MM estimate	Albrecht MM error (%)
1	287.00	613.67	113.82
2	82.50	81.90	−0.73
3	1,107.31	1,103.76	−0.32
4	86.90	515.85	493.61
5	336.30	1,103.51	228.13
6	84.00	105.61	25.72
8	130.30	479.91	268.32
9	116.00	702.00	505.18
11	258.70	615.16	137.79
12	230.70	301.74	30.79
13	157.00	383.59	144.33
14	246.90	392.08	58.80
Mean (absolute values)	260.30	533.23	167.29
Standard deviation	280.60	327.25	177.07

TABLE VIII. Function Points—COBOL KSLOC Data

Project number	KSLOC	Albrecht KSLOC estimate	Albrecht error (%)
1	253.60	137.98	−45.59
2	40.50	53.73	32.66
3	450.00	267.33	−40.59
4	214.40	87.10	−59.37
5	449.90	152.28	−66.15
6	50.00	43.52	−12.96
8	200.00	111.38	−44.31
9	289.00	182.59	−36.82
11	254.20	184.74	−27.33
12	128.60	87.16	−32.22
13	161.40	75.52	−53.21
14	164.80	153.46	−6.88
Mean (absolute values)	221.37	128.07	38.17
Standard deviation	131.14	64.65	17.47

well with eventual SLOC. The results for the ABC were as follows:

ABC Cobol KSLOC $\qquad \bar{R}^2$
$= -5.0 + 0.20$ (Function Points) $\qquad 65.6\%$
$\quad (4.69)$

ABC Cobol KSLOC
$= -13.2 + 0.207$ (Function Counts) $\qquad 75.1\%$
$\quad (5.85)$

Although the IBM and ABC models are apparently fairly similar, the results of an *F*-test are that we can reject the null hypothesis that the models are the same at the 99 percent level.

The second result is that, for the ABC data, the unmodified Function Counts have a higher correlation than the modified Function Points. This suggests that the functionality represented by Function Counts is related to eventual SLOC, but that the 14 "complexity adjustment" factors are not adding any information for this particular sample.

5.5 ESTIMACS Results

As explained in the "Methodology" section (Section 4), the procedure for testing ESTIMACS was to gather the data (excluding the actual man-months), send them to MACS, and receive the estimate. This procedure was performed on 9 of the 15 projects in the database. These 9 were simply the first projects for which data were collected. The data for ESTIMACS are presented in Table IX.

The ESTIMACS average error is 85 percent, which includes some over and some under estimates, and is the smallest average error of the four models. This may be due to the similarity of the ESTIMACS application base (originally an insurance firm) and the ABC database, although this cannot be verified due to the proprietary nature of the ESTIMACS data. The regression data were as follows:

Actual man-months $\qquad \bar{R}^2$
$= 31 + 0.723$ (ESTIMACS) $\qquad 13.4\%$
$\quad (1.50)$

Although the average error was less, the fit with the actual effort is worse than the other models. The *t* statistic is also less, although still significant at approximately the 92 percent confidence level.

To check the possibility that the 9-project subset estimated by ESTIMACS was somehow different from the 15-project data set as a whole, each of the models and its inputs were run against only that 9-project data set. The results were the following:

Actual man-months $\qquad \bar{R}^2$
$= 54.1 + 0.084$ (SLIM) $\qquad 92.8\%$
$\quad (10.17)$

Actual man-months
$= 11.4 + 0.195$ (COCOMO-Basic) $\qquad 93.6\%$
$\quad (10.82)$

Actual man-months
$= 51 + 0.17$ (COCOMO-Intermediate) $\qquad 94.9\%$
$\quad (12.24)$

Actual man-months
$= 40.3 + 0.181$ (COCOMO-Detailed) $\qquad 94.4\%$
$\quad (11.62)$

Actual man-months
$= -203 + 0.451$ (Function Points) $\qquad 71.8\%$
$\quad (4.63)$

Actual man-months
$= -175 + 0.426$ (Function Count) $\qquad 66.2\%$
$\quad (4.09)$

TABLE IX. ESTIMACS Data

Project number	Acutals MM	ESTIMACS	ESTIMACS error (%)
1	287.00	230.26	−19.77
2	82.50	111.26	34.86
3	1,107.31	523.51	−52.72
4	86.90	234.59	169.95
5	336.30	687.63	104.47
8	130.30	389.26	198.74
10	72.00	67.20	−6.67
11	258.70	624.43	141.37
12	230.70	324.78	40.78
Mean (absolute values)	287.97	354.77	85.48
Standard deviation	322.48	219.66	70.36

Actual man-months
$= -94 + 1.67$ (KDSI) 44.5%
(2.90)

Actual man-months
$= -102 + 1.68$ (KSLOC) 43.2%
(2.83)

It does not appear that this 9-project subset was any more difficult to estimate than the full 15-project data set. In fact, SLIM, Function Points, and all three versions of COCOMO did better on the subset. Only KDSI and KSLOC did slightly worse, and these are not inputs to ESTIMACS. If anything, this suggests that the 6 projects not estimated by ESTIMACS were more difficult to accurately estimate than the original 9.

Another possibility may be related to how the input data for the ESTIMACS analysis were collected. Of the 25 required data items, 5 deserve special mention: (1) number of major business functions, (2) number of unique business inputs, (3) number of unique business outputs, (4) number of logical files, and (5) number of on-line inquiry types. These factors (which ESTIMACS collectively refers to as "volume") tend to drive most estimates.[8] Upon examining the ESTIMACS documentation, the last four of these questions sound exactly like the 1979 version of Albrecht's Function Points. For example, here are two direct quotations on the subject of inputs, the first by Albrecht and the second by Rubin:

> *External Input Types*: Count each unique user data or user control input type that enters the external boundary of the application being measured, and adds or changes data in a logical internal file type. An external input type should be considered unique if it has a different format, or if the external design requires a processing logic different from other external input types of the same format. . . . Do not include the input part of the external inquiry types as external input types, because these are counted as external inquiry types. [3]

> *Number of External Inputs*: How many unique logical business inputs will the system process? This is a count of the number of major data types that will enter the system from outside of it. In this way internal data base information is excluded from this total. On-line inquiry only screens are also excluded. Think in terms of major business communication transactions. Inputs counted should be unique in that they require different processing logic from other inputs. [22]

These two questions appear to address the same concept, and Albrecht's 1979 paper is cited in Rubin's 1983 paper. Therefore, in conducting this research, the Function Point definitions were used to answer the matching ESTIMACS questions. According to the MACS staff, however, this may result in overestimating these input quantities by a factor of three[9] or four.[10] How the difference is defined was not available from MACS, so "correct" use of the model may come only with experience. This begets the question of whether the use of Function Points for the "Function-Point-like" questions affected the model's performance. If the estimates had been uniformly high, then this would seem a likely possibility. However, given that the estimate errors ranged from −52.7 percent to +198.7 percent, this does not seem to be the case.

5.6 Sensitivity Analysis

As discussed in Section 3, this data set of 15 completed software projects is large relative to many other published studies. However, it is still small in terms of what researchers might like in order to assure that the results are robust. In particular, readers should note the sensitivity of the regression models to project number 3, which, although almost exactly equal to project number 5 in its KSLOC count, is several times larger in its actual man-months figure. Were this project to be dropped from the data set, almost all of the regression models on the new 14-point data set (particularly COCOMO's) would exhibit decreases in their $\bar{R}^2$, the measure of "goodness of fit". The sole exceptions to this were the KDSI and KSLOC regressions, which exhibited very slight increases. This last result is likely due to the elimination of one of the two similarly sized projects (450 KSLOC) that had very different productivity rates. This would tend to make the remaining large project somewhat of an "anchor" for the regression.

A similar analysis was done for the 9-project "ESTIMACS subset." The results of the regressions run on the 8-project subset were that the $\bar{R}^2$ for SLIM, COCOMO, and ESTIMACS were all in the 38–49 percent range. The results for Function Points and Function Counts decreased to the 54–60 percent range, and the results for KSLOC and KDSI increased to the 56–59 percent range. Again, this last result is likely due to the elimination of one of the two similarly sized projects (450 KSLOC) that had very different productivity rates. As a test of this, the data were rerun on a 7-project subset without either of the largest projects. The models whose performance improved with the 8-project data set (ESTIMACS, KSLOC, KDSI) all worsened as a result. Therefore, the improvements seemed only related to the creation of project number 5 as a new "anchor." Consequently, project number 3 was not dropped, given

[8] Personal correspondence, P. Madden, July 1985.

[9] Personal correspondence, L. Kleeck, 7/18/85.

[10] Personal correspondence, P. Madden, 8/13/85.

no a priori reason for doing so, plus the new regression models' increased dependence on project number 5, the "new outlier" if project number 3 were dropped.

Owing to large differences in project productivity, variation such as the difference between project number 3 and project number 5 is not uncommon in project data. As an example, Albrecht's 24-project data set contains the following two projects:

	KSLOC	Man-Hours (in thousands)
Project number 1	130	102.4
Project number 2	318	105.2
24-project mean	61	21.9
24-project standard deviation	62	27.8

The difference between his largest project (number 2) and the next largest (number 1), in terms of KSLOC, is 188, which is a difference of 3.03 standard deviations. The equivalent calculation for the ABC data is 2.93 standard deviations. The difference between Albrecht's largest project and the mean is 257, or 4.15 standard deviations, whereas the equivalent calculation for ABC is 3.38. Therefore, the ABC data have a less severe outlier problem than even Albrecht's data.

The existence of these outliers and large variations in productivity is not a particularly desirable situation from a research perspective. However, given the difficulty and cost in gathering this kind of data, the best available data are relatively small sets that are subject to this type of condition.

6. CONCLUSIONS

Having examined each model independently, it seems appropriate to summarize these results. Specifically, it is time to address the questions posed in the introduction to this paper to see what answers have been provided by this research. Of course, it will be important to bear in mind that these results stem from 15 projects developed by one firm, and overly broad generalizations are not possible. Rather, these results suggest certain hypotheses that may be verifiable with further research.

6.1 Implications for Practitioners

The first research question to be answered concerned the accuracy of the models outside their original environments and the ease with which they could be recalibrated to fit a new environment. One conclusion is that models developed in different environments do not work very well uncalibrated, as might be expected. Average error rates calculated using the MRE formula ranged from 85 to 772 percent, with many in the 500–600 percent range. This variation is most likely due to the degree to which the productivity of the environments where the models were developed matches the target (ABC) environment. This points out the need for organizations that wish to use algorithmic estimating tools to collect historical data on their projects in order to calibrate the models for local conditions.

After allowing for calibration, the best of the models explain 88 percent of the behavior of the actual man-month effort in this data set. This result alone probably justifies consideration of an algorithmic estimating method by a software manager who has sufficient data to calibrate it to his or her environment. In addition, there are benefits to the estimation task simply from the structure the models impose on the estimation process. Providing inputs for model parameters and sizing estimates requires a project team to carefully consider many important aspects of the upcoming project. Of course, as the model developers themselves point out, these models are adjuncts to, not substitutes for, a detailed estimate by task by the project management [3].

The second research question concerned the relative efficacy of SLOC models versus non-SLOC models. In terms of the MRE results, the non-SLOC models (Function Points and ESTIMACS) did better, although this is likely due to their development in business data-processing environments similar to ABC's. In terms of the regression results, both of the SLOC models (COCOMO and SLIM) had higher correlations than either ESTIMACS or Function Points. However, these data must be considered in light of an important implementation question. The SLOC counts were obtained *ex post* and are therefore likely to be much more accurate than SLOC counts obtained before a project begins. Although SLIM does provide a means for modeling this initial uncertainty (through a beta distribution approach), it is unlikely that a project manager could estimate SLOC counts with the degree of accuracy used in this research. Presumably, the variance between *ex ante* and *ex post* Function Point counts is less, although verifying this is an open research question.

An additional consideration is that Function Points and Function Counts can be used as predictors of KSLOC. In particular, Function Counts correlated with KSLOC at the level of 75.1 percent, which is similar to the correlations published by Albrecht [3], and is likely to be good enough to be of use to the software manager.

The third and final research question concerns the relation between the proprietary and the nonproprietary models. This question was not answered conclusively by this research, as the proprietary

SLIM model outperformed (in its $\bar{R}^2$ value) the nonproprietary COCOMO model on this data set, while the nonproprietary Function Points model outperformed the proprietary ESTIMACS model on this data set.

6.2 Directions for Future Research

This paper has provided several important research results regarding software metrics and models. First, Albrecht's model for estimating man-months of effort from Function Points has been validated on an independent data set. This is particularly significant in that Function Points has been proposed by IBM as a general productivity measure and prior to this there was only limited evidence for their utility from non-IBM sources [4].

One interesting research question raised by this Function Points result is the degree to which *ex ante* and *ex post* Function Points counts are similar. As pointed out in the previous section, the higher correlations generated by the SLOC-based models must be tempered by a recognition that, in practice, accurate *ex ante* SLOC estimates may be difficult to achieve. An interesting research question involves the degree to which initial misestimates change the results achieved by these models. A more general extension of this is, what impact do the estimates themselves have on projects? A recent paper by Abdel-Hamid and Madnick [1] discusses this issue from a systems dynamics perspective. The authors make the point that the effort required on a software-development project is in part a function of the estimate given at the beginning of the project. From this premise they devise a model (not shown in the cited paper) that presumably includes the estimate as an endogenous variable. They then develop an example to show that an estimate that was more accurate could also be more expensive in terms of creating a bigger project.

Although an interesting example, one problem with it is that they draw the unsupported conclusion that the accuracy of software cost estimation models cannot be judged by their performance on historical (completed) projects. Clearly, *ex post* estimates cannot influence the project. Therefore, it seems reasonable to evaluate them by this method. It could be argued that software cost estimation models that do not include their own estimate are deficient, but this is a different conclusion than the one drawn by the authors. However, their idea of the effect of the estimate on a project is an interesting one, and further research should be done in this area.

Finally, although improving estimation techniques within the industry is a worthwhile goal, the ultimate question must concern how the productivity of software developers can be improved. These estimation and productivity questions are related in that the estimation models contain descriptions of what factors their developers believe affect productivity. How well do these models identify and reflect these factors?

The models researched in this study do not seem to capture productivity factors very well. There are several pieces of evidence to support this conclusion. The first is that the COCOMO–Intermediate and COCOMO–Detailed Models (the versions with the productivity factor "cost drivers") did not perform significantly better than the Basic Model and in fact correlated less well with actual man-months. Second, the raw Function Count data correlated as well as or better than the Function Point numbers, which reflect the 14 "processing complexity" adjustments. Third, ESTIMACS, with its 20 additional productivity-related questions, did less well than the similar Function Counts alone. Finally, the model that had the highest correlation, SLIM, has the problem that its questions designed to elicit productivity-related information uniformly generated PFs that were too low (the reason all the default estimates were too large) and MBIs that were too high. This was determined by comparing these values to the values that would have been obtained had the SLIM "calibrate" feature been used.

A reasonable conclusion from all this is that the models, although an improvement over their raw inputs for estimating project effort, do not model the factors affecting productivity very well. One possible extension of this research is to analyze the data in this study to attempt to determine the causes for the wide swings in accuracy of the estimates across projects. What characteristics make a project amenable to this type of estimation? What factors could be added to the models to enable them to do a better job on all of the projects? On the productivity side, the projects in this data set show a large amount of variance in terms of such traditional metrics as SLOC per man-month. Can these variations be traced to productivity factors controllable by the software manager? What are the effects of modern programming practices, such as the use of fourth-generation languages, or AI techniques? Further research needs to be done to isolate and measure these factors affecting systems professionals' productivity if the profession is to meet the challenges of the future.

Acknowledgments. The author wishes to thank Fred Forman, Charles Kriebel, and Mary Shaw for their assistance with this paper. Additional thanks to Douglas Putnam of Quantitative Software Management and Patrick Madden of Management and Computer Services for permission to use their models during this research.

REFERENCES

1. Abdel-Hamid, T., and Madnick, S. Impact of schedule estimation on software project behavior. *IEEE Softw.* 3, 4 (July 1986), 70–75.
2. Albrecht, A.J. Measuring application development productivity. In *Proceedings of the IBM Applications Development Symposium, GUIDE/SHARE* (Monterey, Calif., Oct. 14–17). IBM, 1979, pp. 83–92.
3. Albrecht, A.J., and Gaffney, J., Jr. Software function, source lines of code, and development effort prediction: A software science validation. *IEEE Trans. Softw. Eng. SE-9*, 6 (Nov. 1983), 639–648.
4. Behrens, C.A. Measuring the productivity of computer systems development activities with Function Points. *IEEE Trans. Softw. Eng. SE-9*, 6 (Nov. 1983), 648–652.
5. Boehm, B.W. *Software Engineering Economics*. Prentice-Hall, Englewood Cliffs, N.J., 1981.
6. Brooks, F.P. *The Mythical Man-Month*. Addison-Wesley, Reading, Mass., 1975.
7. Callisen, H., and Colborne, S. A proposed method for estimating software cost from requirements. *J. Parametrics 4*, 4 (Dec. 1984), 33–40.
8. Conte, S., Dunsmore, H., and Shen, V. *Software Engineering Metrics and Models*. Benjamin/Cummings, Menlo Park, Calif., 1986.
9. Ferens, D.V. Software support cost models: Quo vadis? *J. Parametrics 4*, 4 (Dec. 1984), 64–99.
10. Gaffney, J.E., Goldberg, R., and Misek-Falkoff, L. SCORE82 Summary. *Perform. Eval. Rev. 12*, 4 (Winter 1984–1985), 4–12.
11. Golden, J.R., Mueller, J.R., and Anselm, B. Software cost estimating: Craft or witchcraft. *Database 12*, 3 (Spring 1981), 12–14.
12. Jones, C. *Programming Productivity*. McGraw-Hill, New York, 1986.
13. Kitchenham, B., and Taylor, N.R. Software cost models. *ICL Tech. J. 4*, 1 (May 1984), 73–102.
14. Masters, T.F., II. An overview of software cost estimating at the NSA. *J. Parametrics 5*, 1 (Mar. 1985), 72–84.
15. Pinsky, S.S. The effect of complexity on software trade off equations. *J. Parametrics 4*, 4 (Dec. 1984), 23–32.
16. Putnam, L.H. General empirical solution to the macro software sizing and estimating problem. *IEEE Trans. Softw. Eng. SE 4*, 4 (July 1978), 345–361.
17. Putnam, L., and Fitzsimmons, A. Estimating software costs. *Datamation 25*, 10–12 (Sept.–Nov. 1979).
18. Quantitative Software Management. *Reference Notes for the DOD SLIM Software Cost Estimating Course*. Quantitative Software Management, McLean, Va., 1983.
19. Quantitative Software Management. *SLIM User Manual (IBM PC Version)* Draft copy ed. Quantitative Software Management, McLean, Va., 1984.
20. Rubin, H.A. Macroestimation of software development parameters: The Estimacs system. In *SOFTFAIR Conference on Software Development Tools, Techniques and Alternatives* (Arlington, Va., July 25–28). IEEE Press, New York, 1983, pp. 109–118.
21. Rubin, H.A. The art and science of software estimation: Fifth generation estimators. In *Proceedings of the 7th Annual ISPA Conference* (Orlando, Fla., May 7–9). International Society of Parametric Analysts, McLean, Va., 1985, pp. 56–72.
22. Rubin, H.A. Using ESTIMACS E. Management and Computer Services, Valley Forge, Pa., Mar. 1984.
23. Software Productivity Research. *SPQR/20 User Guide*. Software Productivity Research, Cambridge, Mass., 1986.
24. Theabaut, S.M. Model evaluation in software metrics research. In *Computer Science and Statistics: Proceedings of the 15th Symposium on the Interface* (Houston, Tex., Mar.). 1983, pp. 277–285.
25. Walston, C.E., and Felix, C.P. A method of programming measurement and estimation. *IBM Syst. J. 16*, 1 (Jan. 1977), 54–73.
26. Weisberg, S. *Applied Linear Regression*. Wiley, New York, 1980.
27. Wolverton, W.R. Cost of developing large scale software. *IEEE Trans. Comput. C-23*, 6 (June 1974), 615–634.

CR Categories and Subject Descriptors: D.2.8 [**Software Engineering**]: Metrics—*performance measures*; D.2.9 [**Software Engineering**]: Management—*cost estimation*; K.6.1 [**Management of Computing and Information Systems**]: Project and People Management—*management techniques*; K.6.3 [**Management of Computing and Information Systems**]: Software Management—*software development*

General Terms: Management, Measurement, Performance

Additional Key Words and Phrases: COCOMO, ESTIMACS, Function Points, SLIM, SLOC

Received 3/86; revised 8/86 and 10/86; accepted 10/86

Author's Present Address: Chris F. Kemerer, Graduate School of Industrial Administration, Carnegie-Mellon University, Schenley Park, Pittsburgh, PA 15213.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.