A Comparison of Parametric Software Estimation Models Using Real Project Data

George Stark, IBM Global Services

Section 1: Introduction

Defense project managers and software engineers are often called upon to produce effort, duration, and quality estimates for a new project based on the project's initial needs statement. Often the manager or engineer is solely responsible and accountable for producing and delivering these estimates; in other cases, a senior executive may ask them to develop estimates for his or her use.

Depending on the time available (usually short turnaround–one or two days–is required), the level of uncertainty associated with the project scope (often only a general vision or statement of capability), and the phase of the project (early concept is most common), estimators often rely on one or more rules-of-thumb to arrive at their estimate. Most published models have guidelines for these rules, but there is little empirical data to show how well they work. This paper provides that empirical data for one organization's software development approach.

Project estimation involves translating a set of business objectives or requirements into a measure of product "size." This size measure is then used to estimate the effort, duration, and quality of the final software product. The ability of a system engineer or project manager to align the business objectives with the technical estimates leads to well informed business decisions. The time to complete an estimate is often sufficient only for the use of "rules-of-thumb" for simple models to generate "ballpark" estimates that can then be refined as the project unfolds. Several authors, including Boehm [1,2], Jones [3,4], Rone [5], and others [6-10], have published approaches to arrive at software development effort, project duration in calendar time, and quality as measured by defects discovered prior to release. In all cases the authors advocate for calibrating their approaches based on data from the organization's previous projects … but what if this data is not available? The questions, then, are: (1) How good are these approaches "out of the box" using the parameters from the model author's environments?; and (2) Can we rely on them to make business decisions?

Several authors have contributed critiques and comparison papers of the various models:

• Atkinson, et al. [11] and Pearse, et al. [12] show how simple software metrics, both actual and estimated, can be used to effectively manage the final stages of software development, but they do not address early project estimators.

• Kemerer [13] concluded that metrics-based software project estimation is a viable approach as long as the models are calibrated for the environment. He also concluded that function point-based size estimates were better than source lines of code for the 15 projects studied from one environment.

• Jorgensen and Sheppard [14] found that more than 50% of estimation articles try to build, improve, or compare regression-based estimation models. In further studying expert judgment estimation, they also identified a lack of in-depth studies on the actual use of the approach and real-life evaluations published as journal papers.

• Fenton and Pfleeger [15] concluded that single models may work well in environments for which they were derived, but do not translate well to other environments because of the availability of parameter drivers early on in the estimating process. They recommend changes to estimation model structure and standardization of local data definitions to reduce input subjectivity.

• Jorgensen [16] also reported that expert judgment is the predominant estimation technique used in industry today. He analyzed 15 studies comparing model-based and expert-based effort estimation. The results were a tie: Five in favor of model, five in favor of the expert, and five had no difference. Thus, there was no clearly superior approach to effort estimation.

These results encouraged us to continue to search for a parametric approach that would help us to quickly create a reasonable bound on the effort, duration, and quality for a particular project request. Our investigation identified several candidate models that a systems engineer could apply with little to no historical data. But, just because we could … should we?

The findings presented herein pertain to our particular collection of 54 completed projects by a well-established and measured software development organization. It is possible that some of our findings are not generally applicable, so practitioners are encouraged to run their own tests and determine which of the available models is best for their particular environment. Nevertheless, most of the results presented here are consistent with our intuition and the conclusions above. Thus, we believe the results provide a good perspective of the value of metrics-based software project estimators and the corresponding rules-of-thumb provided by their inventors.

The next section provides an overview of the various estimating models we tested (four Effort prediction models, three Dura-

tion prediction models, and two defect prediction models). This is followed in Section 3 by a description of the projects included in the analysis and the data collected. Section 4 presents the comparisons and the findings.

Section 2: Estimating Models

The following subsections describe the models that we evaluated.

Effort Models

Three common formulas for estimating the effort (in person-months) are based on delivered source lines of code (SLOC). Rone developed a model at IBM based on observations from a variety of software projects for customers, including the space shuttle flight software, school district management, point-of-sale systems, help desk systems, and others. The model has the form:

E = ((((SLOC/productivity)*1.1)*1.2)*1.3)

Where E is the effort measured in person-months, SLOC is the delivered project scope measured in thousands of lines of code, and productivity is measured as lines of code/development person-months spent between design and functional test. The multipliers are for independent test (1.1), systems engineering and architecture (1.2), and project & configuration management and additional overhead (1.3).

Similarly, Bailey and Basili [6] developed a formula based on 18 large FORTRAN projects. It is expressed as:

E = 3.4 + 0.72 * KSLOC^1.17 plus or minus 1 standard deviation

Where effort is measured in person-months, and KSLOC is the delivered project scope measured by thousands of lines of delivered code. While the goal of the article was to demonstrate an approach to calibration using step-wise regression, our question is, "Can this model, with its published parameters help deliver a ready-made estimation formula?"

Finally, Barry Boehm's original COCOMO model [1] has an effort formula for three different types of systems: organic systems, semidetached systems, and embedded systems. The formula is:

E = 2.4 * KSLOC^1.05 (organic systems)
E = 3.0 * KSLOC^1.12 (semidetached systems)
E = 3.6 * KSLOC^1.20 (embedded systems)

Where KSLOC represents the delivered project scope measured by thousand delivered source instructions. Boehm's model has 14 factors that allow the estimator to tailor the estimate by +/- 65% using subjective assessments of each factor. This range was used to establish the upper and lower bounds for the analysis performed.

Caper's Jones [3, 4] and the International Software Benchmarking Standards Group (ISBSG) [7, 8] have also published rules-of-thumb formula to help estimators. These formula are based on project size calculated using Jones' function points metric [17]. The effort estimator is:

Effort = Project Size in Function Points * Productivity

Caper's Jones has derived the staff productivity (measured in function points/staff-hour) as a function of project size, which can be found in [18]. The ISBSG has also empirically derived the productivity rates of project teams developing on various platforms from their project database. These are shown in Table 1.

Table 1: ISBSG Productivity Rates

Platform	10th	Median	90th	Mean
Mainframe	3.2	11.9	34.4	16.8
Midrange	3.8	10.3	30.6	14.1
PC	2.2	7.1	23.1	10.2
Multi	2.6	6.9	22.2	10.7

Duration Models

Three of the previously cited sources also offered simple estimates of project duration. These are summarized in Table 2, where D is the project duration in calendar months, Effort is Staff-months, FP is Function Points, and C is a constant representing upper and lower bounds for the estimate's rule-of-thumb.

Table 2: Duration Rules of Thumb

Source	Rule equation	Parameters
Boehm	$D = 2.5*(Effort)^C$	C = [0.32, 0.38]
Jones	$D = FP^C$	C = [0.36, 0.46]
ISBSG	$D = 0.971*FP^C$	C = [0.35, 0.50]

Quality Models

Both Rone and Jones also offered simple estimates of the quality of the product in terms of the number of defects that should be removed prior to release. The two models are summarized in Table 3, where Q is the expected number of defects, KSLOC is thousands of delivered source lines of code, FP is Function Points, and C is again an empirical constant representing the model's upper and lower bounds.

Table 3: Quality Rules-of-Thumb

Source	Rule equation	Parameters
Rone	Q = KSLOC*C	C = [5.4, 11.2]
Jones	$Q = C*FP^{1.25}$	C = [0.05, 1]

Section 3: Project Data

Fifty-four projects were selected from an organizational repository containing more than 150 projects. In most respects, these projects are typical of an organization assessed to be at Software Engineering Institute (SEI) CMMI® Level 2/3. This means that good practices are followed and that project plans are tailored from the organizational standard process to ensure a good fit between the project goals and activities. Data is collected on all projects and used to track and control the project against the committed plan of record.

The authors had some prior experience with the selected projects and chose the projects from the repository strictly on the basis of the availability of the project data, rather than the data values. The following criteria were used to select the 54 projects analyzed in this study:

• **Completed**. No in-progress projects were included.

• **Recent.** The repository contains projects since 2000. We decided to only use projects completed since the organization demonstrated an SEI CMMI assessment of Level 2 with some organizational characteristics at Level 3. The organization was assessed at Level 3 with some characteristics at Level 4 in 2006.

• **Software-related.** The repository contains hardware and deployment only projects. Since the estimation models are specific to software, we ignored all non-software projects.

• **Life-cycle phase.** For each completed project, the repository includes data from initial project concept through product release. We excluded the concept phase. We included data that went from technical specification through integration test and release.

• **Necessary metrics.** All data items related to size, effort, duration, and quality were available. Some projects in the repository did not record all data items needed to calculate the effort, duration, and quality models, and, thus, could not be used. For example, many projects collected use cases, objects, or story points for the size measure and could not be used with these models.

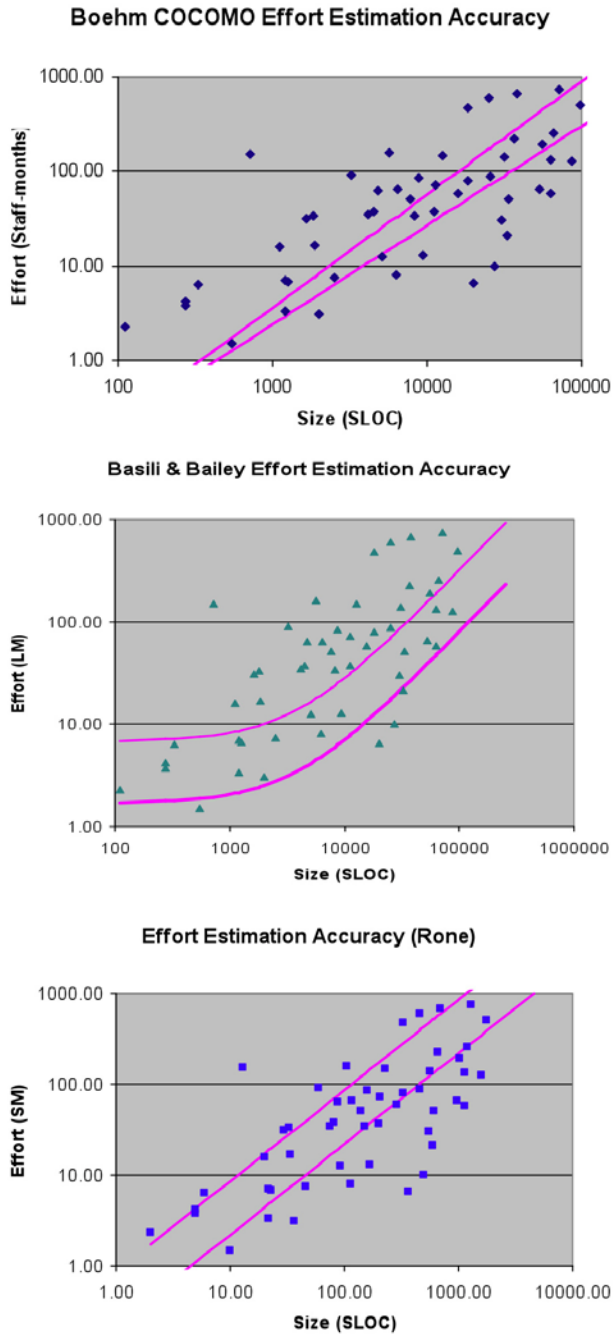
Thus, after eliminating the unfinished, pre-Level 2, non-software, and project missing necessary data items, we were left with 54 measureable software projects representing a cross-section of the organization's business. That is, there are innovative development projects, commercial product integration projects, and maintenance/enhancement projects of varying sizes, durations, and levels of process tailoring. These projects are multi-platform (e.g., AIX, Linux, Windows), multi-language (e.g., C, C++, Java, KSL, PERL and other scripting languages), and multi-disciplinary (e.g., IT monitoring solutions, help desk ticketing systems, telephony systems, banking systems).

Section 4: Results

The following collection of plotted graphs show how each of the project estimation models fit the actual data derived from the projects pulled from the historical repository. Each plotted point represents one of the actual projects. Projects plotted above the upper boundary line required more effort than the model predicted; those below the lower boundary line required less effort than the model predicted. Plots with a grey background SLOC as the project content estimator, while those with a white background use the Function Point approach to measure the project content.

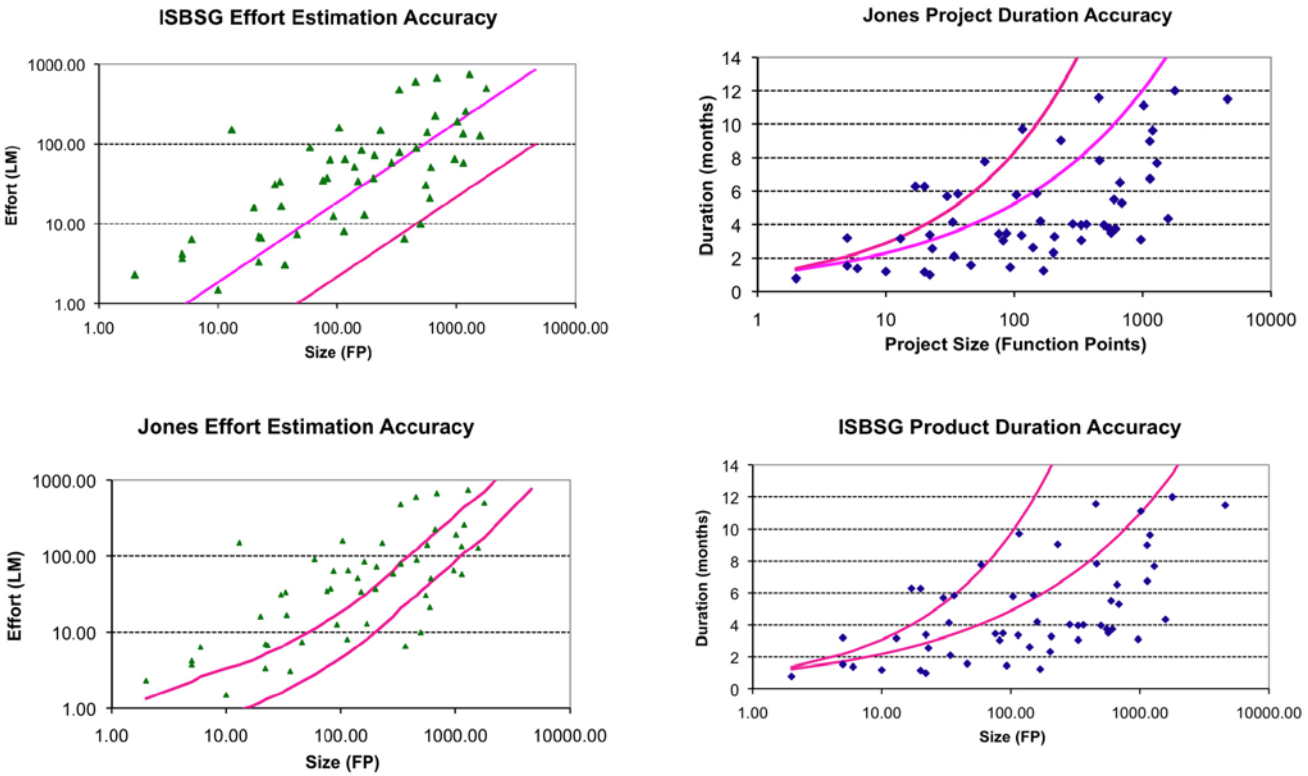
Figure 1 shows the results for the SLOC effort estimation models. The boundaries (for COCOMO I) include the minimum and maximum multipliers for the subjective adjustments available in the model. Using these bounds, only 24% of the projects in our database fell within the bounds of the COCOMO model and more than 46% fell above the upper bound, meaning that the model was very optimistic for our environment. The Bailey and Basili effort estimation model did not fare much better, with 54% of the projects coming in above the upper bound estimate and 33% falling within the bounds. The boundaries on the Rone effort estimation model contained roughly 40% of our projects with 20% of the projects above the upper boundary and the remaining 40% below the lower boundary.

Figure 1: SLOC Effort Estimation models



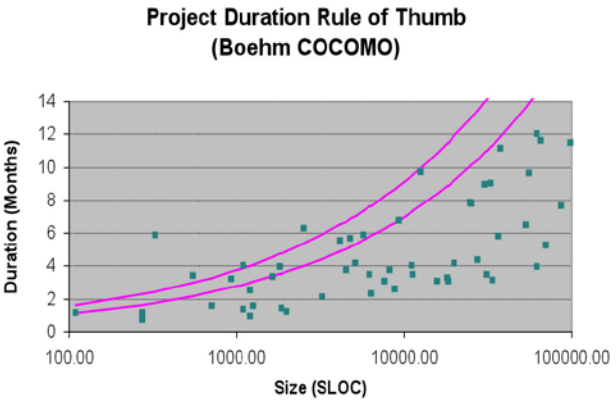
The Function Point effort estimation model results are shown in Figure 2. They did not fare any better than the SLOC models. 50% of our projects were above the Jones upper bound and 63% were above the ISBSG upper bound. In fact, only 13% of the projects were below the lower bound on the Jones model and only 4% of the projects were below the lower bound for the ISBSG approach. This would lead us to conclude that the lower bounds on these rules-of-thumb for effort estimation should never be used in our environment.

Figure 2: Function Point Effort Estimation models



The estimates for duration were significantly better than those for effort. Figure 3 contains the results for those models. Only four of the 54 projects (7%) were above the upper bound associated with the COCOMO-I model, indicating that they took more calendar time than the model estimated. A full 93% were contained below the upper bound and 78% were below the lower bound indicating that it is a cautious estimate to follow early in the process. The ISBSG approach was also quite good with 89% of the projects taking less time than the upper bound on duration using their model and 70% being below the lower bound. The Jones model was comparable with 80% below the upper bound and 75% below the lower bound.

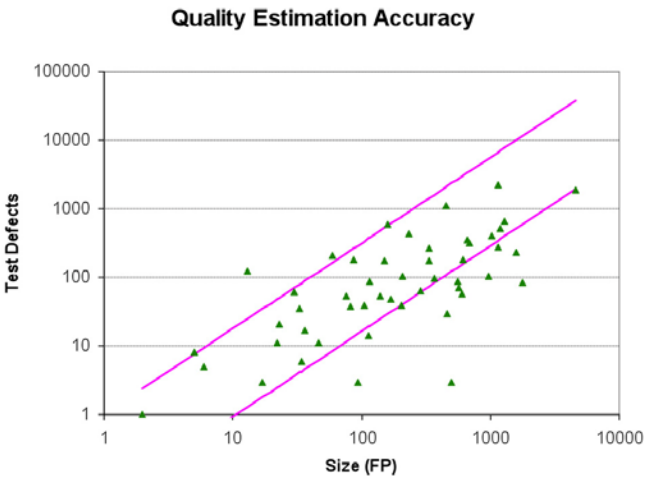
Figure 3: Project Duration Model results



In general, any of these three duration approaches would give our organization a 75% chance of delivering the project on time using the shortest duration estimate.

Figure 4 depicts the results of the quality (i.e., defect) estimates. The Jones model fared well with 60% of the projects falling between the model bounds and 93% of the projects having fewer defects than the model upper bound estimated. The Rone model bounds contained 24% of the projects, with an additional 31% falling above the upper bound estimate.

Figure 4: Quality Estimation Model results



Summary

Defense managers and system engineers require estimates of project cost/effort, duration, and quality in order to secure funding and set expectations with customers, end users, and management teams. Researchers and practitioners of software metrics have developed models to help project managers and system engineers produce estimates of project effort, duration, and quality. These models generally quantify the project scope using estimated source lines of code or function points, and then require the application of generalized rules-of-thumb to arrive at the needed project estimates of staffing, duration, and quality. Experts agree that for these models to perform at their best, the parameters should be calibrated based on project data from the using organization. Our question was, "How do parametric models perform out-of-the-box (that is, without calibration)?" This is analogous to a project team without access to historical data using the models as published. What level of accuracy can they expect? We examined several published models by comparing the predicted values against the actual results from 54 software projects completed by a SEI CMMI Level 3 organization with a mature (and commended) measurement program.

This paper evaluated a subset of these approaches – nine simple models (four effort estimation models, three duration estimation models, and two software quality (i.e., defect) models)– using 54 non-trivial commercial projects completed recently by a CMMI Level 3 organization. This certification means that the data was collected in a standard manner and makes sense to use in this study. It does not imply that a defined process level is needed to use the results.

For the effort estimation models, we found that the upper bound of the best case model contained 81% of our projects, that is, four out of five of our projects would use less effort than predicted by the best case model, whereas the average effort estimate across all models contained only 54% of our projects, or a little better than a coin flip if we estimate using the average.

Duration estimates performed significantly better. In the best case model, the upper bound estimate contained 93% of our projects with the overall model average at 91% and the lower bound estimate exceeded the actual duration more than 70% of the time. This means we can out-perform the project duration seven out of 10 times using the shortest duration estimated using the models out-of-the box.

For quality modeling, one of the defect prediction approaches worked quite well, with the upper bound containing 94% of the projects (or 9.4 times out of 10 we will deliver fewer defects than forecast by the model). This information is useful to executives and managers performing early project estimates without detailed analysis of the requirements or architecture as the bounds allow them to quickly respond to customer requests with some level of confidence.

So, if you are asked for a project estimate and do not have access to historical data or well-calibrated local estimation models, there is hope. Based on your available sizing information, you can use these models out-of-the-box with some success as long as you keep these things in mind:

- Caper's Jones approach was the only one that (relatively) accurately addressed all three project management estimation needs for effort, duration, and quality.
- None of the four effort estimation models were particularly effective with our project data, but using the upper bound of the Rone model gives the project team an 80% chance of meeting the effort estimate.
- A project should never commit to the lower bound effort estimates from any of the models we evaluated.
- The duration estimation models are particularly effective with our project data. Using the upper bound of the Boehm model gives a project team a better than 90% chance of completing the project within the estimated calendar time.
- Capers Jones' quality model was the most accurate predictor of quantity of defects in our software development projects.

From our analysis, it appears as though duration and quality models are quite useful, but effort estimation is still problematic. We suggest researchers investigate other approaches to effort estimation that are not based on SLOC or Function Points. For example, models that rely on use cases or story points and can estimate all three key parameters (i.e., effort, duration, and quality) may prove valuable in the future. The translation from mission or business need to requirements and architecture is a huge challenge that impacts estimates on each iteration, by developing models to address these early solution descriptions, managers and system engineers can benefit with earlier estimates.

Disclaimer:
The opinions and conclusions are those of the author(s) and not of IBM.
® CMMI is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

ABOUT THE AUTHOR

George Stark is an IBM Senior Technical Staff member with over 25 years of experience in software and service measurement and statistical modeling. He has published more than 40 technical papers in referred journals and conferences and was has been on the editorial board of the Software Quality Journal. He is currently a member of the Delivery Excellence team where he consults with IBM quality and productivity improvement teams worldwide and is a key leader in the IBM Estimation Community of Practice. He also works with clients on project estimation, software reliability and process improvement approaches.

George Stark
10033 Circleview Drive
Austin, Tx 78733
(512) 653-5438 phone
(512) 263-5024 fax
gstark@us.ibm.com

REFERENCES

1. Boehm, B. (1981). Software Engineering Economics. Englewood Cliffs, N.J., Prentice Hall.
2. Boehm, B. (2006). "Minimizing Future Guesswork in Estimating," IBM Conference on Estimation, Atlanta, Ga. Feb. 2006.
3. Jones, C., (2007), "Software Estimating Rules-of-Thumb," <http://www.compaid.com/caiinternet/ezine/capers-rules.pdf>, Mar. 2007.

REFERENCES - CONTINUED

4. Jones, C., (1997), Applied Software Measurement, 2nd Ed., McGraw-Hill, NY.
5. Rone, K., et al, (1994), "The Matrix Method of Software Project Estimation", proceedings of the Dual-Use Space Technology Conference, NASA Johnson Space Center, Houston, TX, Feb.
6. J.W. Bailey and V.R. Basili, "A Meta-Model for Software Development and Resource Expenditures," Proceedings of the 5th International Conference on Software Engineering. New York: Institute of Electrical and Electronic Engineers, 1983.
7. ISBSG, International Software Benchmarking Standards Group, <http://www.compaid.com/caiInternet/ezine/ISBSGestimation.pdf>
8. ISBSG, International Software Benchmarking Standards Group, <http://www.isbsg.org/isbsg.nsf/weben/Project%20Duration>
9. McConnell, S., (2006), Software Estimation: Demystifying the Black Art, Redmond, WA, Microsoft Press.
10. P. Oman, "Automated Software Quality Models in Industry," Proceedings of the Eighth Annual Oregon Workshop on Software Metrics, (May 11-13, Coeur d'Alene, ID), 1997.
11. G. Atkinson, J. Hagemeister, P. Oman & A. Baburaj, "Directing Software Development Projects with Product Measures," Proceedings of the Fifth International Software Metrics Symposium, (Nov. 20-21, Bethesda, MD), IEEE CS Press, Los Alamitos, CA, 1998, pp. 193-204.
12. T. Pearse, T. Freeman, & P. Oman, "Using Metrics to Manage the End-Game of a Software Project," Proceedings of the Sixth International Software Metrics Symposium, (Nov. 4-6, Boca Raton, FL), IEEE CS Press, Los Alamitos, CA, 1999, pp. 207-215.
13. Kemerer, C. F., (1987), "An empirical validation of software cost estimation models," Communications of the ACM, Vol 30, No 5, pp. 416-429.
14. Jorgensen, M, and Sheppard, M., (2007), "A Systematic Review of Software Development Cost Estimation Studies," IEEE Transactions on Software Engineering, Vol 33, No 1, Jan. pp 33-53.
15. Fenton N. E., and Pfleeger, S. L., (1997), Software Metrics: A Rigorous & Practical Approach, 2nd Ed., London, PWS Publishing.
16. Jorgensen, M., (2004), "A Review of Studies on Expert Estimation of Software Development Effort," Journal of Systems & Software, Vol 70, no 1, pp. 37-60.
17. International Function Point User's Group (IFPUG), Function Point Counting Manual, Release 3.1, 1990.
18. Jones, C., "Achieving Excellence in Software Engineering," presentation to IBM Software Engineering Group, March, 2006.