Software cost estimating for CMMI Level 5 developers

Corinne C. Wallshein a,∗, Andrew G. Loerch b,1

a Naval Center for Cost Analysis, Washington, DC 20350-1000, United States
b George Mason University, Fairfax, VA 22030, United States

Article history:
Received 8 December 2014
Revised 17 February 2015
Accepted 19 March 2015
Available online 30 March 2015

Keywords:
Software effort estimation
CMMI Level 5
US DOD
Ordinary least squares regression

Abstract
This article provides analysis results of Capability Maturity Model Integrated Level 5 projects for developers earning the highest level possible, using actual software data from their initial project estimates. Since there were no measures to verify software performance, this level was used as a proxy for high quality software. Ordinary least squares regression was used to predict final effort hours with initially estimated variables obviates the need to estimate growth or shrinkage for typical changes occurring in software projects, regardless of software developer (contracted or in-house). The OLS equations, or cost estimating relationship equations, were evaluated by a series of standards: statistical significance, visual inspection, goodness of fit measures, and academically set thresholds for accuracy measures used in software cost estimating: mean magnitude of relative error and prediction (for determining the percentage of records with 25% or less, based on their magnitude of relative error score). As several initial estimated variables were strongly correlated to the reported final effort hours and each other, each variable was examined separately. Thirty records from software projects completed in 2003–2008 for the highest process maturity level were used to compute statistically significant equations with implicit growth or shrinkage in their make-up.

1. Introduction

Software effort estimates provide a basis for funding and budgeting decisions. Software cost can be approximated by taking total estimated effort hours multiplied by average labor costs per hour. Once actual effort occurs, accuracy of estimates can be determined, if initial estimates are available.

Software effort and duration equations in commercial and academic software cost models use a variety of factors to estimate software costs, including: size, complexity, productivity, planned schedule, personnel experience, personnel numbers, requirements volatility, and computing constraints. Additional parameters such as application type, domain, development method, programming languages, and commercial-off-the-shelf (COTS) integration have joined a growing set of hypothesized effort and duration drivers. When software cost model parameters are unknown, many analysts use the contractor’s or in-house team’s bid and past history as parameters for estimating costs. If the past performance came in under bid, the current bid is deflated to estimate costs. Likewise, if the past performance came in over bid, the current bid is inflated to estimate costs.

To improve cost estimation, the Department of Defense (DoD) collects data on software efforts for a variety of major defense programs. This growing collection of programmatic and software data, available to government cost analysts, is intended to be the basis of software cost estimating relationships (CERs) and cost analysis tools. A key data source is the repository of Software Resources Data Reports (SRDRs), where initially estimated software project information can be examined along with the contractor’s delivered actual software project information.

2. Motivation and background

The United States Government Accountability Office (GAO) studied high visibility, large DoD programs’ total research, development, test, and evaluation costs and found systemic growth, averaging 40%, from their original estimates (GAO, 2008). GAO postulated that using disciplined system engineering and software practices (like those in organizations with high CMMI levels) may promote executable business cases and dampen overly optimistic requirements (GAO, 2008). Overly optimistic requirements generally lead to overly optimistic cost estimates. The GAO did not report on differences in software project outcomes based on the developer CMMI levels.

Although many metrics are collected on major DoD programs, quality metrics are sparse. SRDR lists the most recently achieved CMMI level rating which is based on the software process, not the software end product. A key assumption in this research is that
developers with the highest level of process maturity, Level 5, produce high-quality software. Higher levels of process maturity reflect a continuous focus to reduce defects, achieve stakeholder satisfaction, and enhance software management, maintenance and improvement efforts (CMMI Institute web site, 2015).

Many software cost models exist, including the suite of constructive cost models (COCOMO) with an input scaling parameter for process maturity (PMAT) (Boehm et al., 2000). Data from 37 projects in four CMMI Level 5 organizations showed “a steep reduction in variance” in effort and cycle time, lessening the significance of other software modeling factors such as personnel capability and requirements specifications (Agrawal and Chari, 2007). Data from 161 projects showed increasing process maturity with the PMAT parameter in COCOMO II had an inverse effect on development effort (Clark, 2000). Errors in scaling factors, which include PMAT, seem to have less effect than errors in software size and size multipliers (Musilek et al., 2002). Additionally, simple models are prone to less error when estimating factors are derived at the initial analysis stage (Wu, 2006).

This paper aggregates projects from multiple CMMI Level 5 organizations. Data was compared to subsets from lower CMMI levels which did not demonstrate robust statistical relationships as the number of records per level was small and the scatter in the data was large. According to Agarwal and Chari, software size is the most significant variable to predict development effort and cycle time (Agrawal and Chari, 2007). Although software size reigns as a leading independent variable, it is an estimate until after the software is written and counted. Effort estimates have been shown to be highly sensitive to inaccuracy in size estimates (Musilek et al., 2002). Using software size and a base productivity rate, software project managers frequently use algorithmic strategies to estimate development effort and time since abstraction impacts the size count. Tables in the following links, accessed in 2009, were used to normalize software size to logical source code:

- [http://software.gsfc.nasa.gov/docs/QSM-class/Day%201-Cost/04a-Size.ppt](http://software.gsfc.nasa.gov/docs/QSM-class/Day%201-Cost/04a-Size.ppt)

![Fig. 1](http://csse.usc.edu/csse/event/2009/COCOMO/presentations/Workshop%20Summary%20%20Metrics%20Unification%20%20Productivity%20Domains.ppt)

**Fig. 1.** Traditional software cost estimating equation.

The highest possible count assigns 100% to new and 0% to modified and unmodified lines; the largest assigns 100% each to new, modified, and unmodified lines (Gallo et al., 2009).

3. Problem statement

Many DoD cost analysis professionals use the current contract or historical contract performance as a basis for estimating future project costs, regardless of CMMI level and regardless of factors other than the original bid. The choice of top CMMI levels were a conscious choice as a surrogate for quality software (Wallshein and Loerch, 2010). DoD currently does not have a central repository of software defect rates by project or program.

This research extracted initial estimates of software parameters and analyzed whether they could predict final, actual effort using simple, one parameter models. Estimated parameters at the onset were matched to actual effort recorded at the project’s conclusion. To test the proposed effort-estimating relationships, correlation, visualization, fit statistics, and two accuracy metrics were applied. The first metric, mean magnitude of relative error (MMRE), calculates average magnitude of relative error (MRE) for each data point. MMRE values less than 25% are considered to be more accurate. The second metric is prediction (PRED) accuracy. This measure is computed from a summation and a division. Each data point’s MRE is compared against a threshold (such as MRE less than or equal to 0.25). If the data point at least meets the threshold, it is assigned a value of one; otherwise it is assigned a value of zero. The series of ones and zeroes are summed and this sum is divided by the total number of data points to get a percentage. With PRED, if 75% of the predicted values are within 25% of the actual values, then PRED is considered within an acceptable range of accuracy. Desirable MMRE and PRED thresholds were first published in 1986 (Conte et al., 1986).

4. Research methodology

To determine whether collected DoD data fulfills its intended purpose to improve software cost estimating, the research followed a step-by-step process. The first steps were to analyze the data using statistical techniques, such as statistical visualization. The next steps of the methodology were to create software effort estimating relationships and evaluate their performance using multiple criteria.

Data analysis, including statistical visualization of the relationships between initial estimated software parameters and final effort hours, focused on identifying highly correlated initial parameters to final effort hours with ex post facto analysis of secondary data sets. Software records (SRDRs) for contractors rated at CMMI Level 5 contained potential initially estimated independent variables with final software effort hours as the dependent variable. At the time of the data collection period, there were thirty records at CMMI Level 5 satisfying the repository business rules. There were also thirty-four records at CMMI Level 4, and fewer than fifteen at other, lower CMMI levels, analyzed similarly with dissimilar results.

Software size was normalized to logical statements by languages as suggested by Lum et al. with conversion ratios published online (Lum et al., 2008). The lowest possible KESLOC count is estimated new source lines of code in thousands, abbreviated KNEW. This was used as a separate independent variable for the lower bound of KESLOC. For the upper bound of KESLOC, total source lines of code in thousands (KSQLC) was used as a comparative, separate independent variable (Galfo et al., 2009).

Pearson and Spearman Rho correlation matrices determined relationships among the data sets and highlighted potential independent variables which were subsequently analyzed. After analyzing relationships among the data, likely independent variables for this analysis were identified. These likely independent variables were estimated peak staff, estimated effort hours, and estimated software size.
There is a monotone relationship between final, actual effort hours and initial, estimated effort hours. Estimated peak staff also relates monotonically. The monotone relationship is stronger for KSLOC than KNEW in CMMI Level 5. The strongest monotone correlation is peak staff to final effort hours in CMMI Level 5. Software size correlates to KNEW in CMMI Level 5. The strongest monotone correlation is peak staff to estimated effort hours (K).

Overall Spearman Rho correlations were stronger than the Pearson correlations. In the Spearman Rho matrix, CMMI Level 5 correlations were above 0.75 for each proposed independent variable. For the Pearson correlation matrix, the weakest relationship was between KNEW and final actual effort hours. When KSLOC was related to final effort hours, the Pearson correlation was at 0.71, compared to 0.51 for KNEW. The strongest correlation was between estimated peak staff and final effort hours with the Pearson correlation of 0.79 and the Spearman Rho correlation at 0.82. The next strongest Spearman Rho correlation was between estimated effort hours (K) and estimated peak staff value; estimated peak staff values correlate to estimated unmodified KSLOC as well as to estimated software requirements.

Spearman Rho correlation for CMMI Level 5 highlights the monotone relationship of the potential independent variables to actual, final effort hours (K). These relationships for estimated effort hours (K), estimated peak staff, and estimated new source lines of code (SLOC) in thousands (or KSLOC) abbreviated KNEW, and estimated unmodified KSLOC were individually correlated above the ±0.50 threshold to actual, final effort hours (K). Relationships between the variables were examined and the following was observed: estimated effort hours (K) correlate to estimated peak staff value; estimated peak staff values correlate to estimated unmodified KSLOC as well as to estimated software requirements.

For CMMI Level 5, estimated peak staff is the independent variable. Actual hours, divided by thousands of hours, is the dependent variable and the final, actual effort hours. With the raw data or after applying data transformations, all CERs in Table 1 were statistically significant.

For CMMI Level 5, estimated peak staff is the independent variable. Actual hours, divided by thousands of hours, is the dependent variable. Graphs for the standardized residuals indicate they may be skewed data. All regressions were analyzed using the assessment criteria.
Table 1

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Standard error</th>
<th>CV</th>
<th>( R^2 )</th>
<th>( p ) value</th>
<th>MMRE</th>
<th>PRED(25)</th>
<th>Durbin–Watson</th>
</tr>
</thead>
<tbody>
<tr>
<td>EstPS</td>
<td>41.807</td>
<td>0.554</td>
<td>0.617</td>
<td>&lt;0.001</td>
<td>1.531</td>
<td>0.467</td>
<td>1.315</td>
</tr>
<tr>
<td>LN (KNEW)</td>
<td>44.657</td>
<td>0.592</td>
<td>0.682</td>
<td>&lt;0.001</td>
<td>0.585</td>
<td>0.367</td>
<td>1.470</td>
</tr>
<tr>
<td>LN (KSLOC)</td>
<td>60.730</td>
<td>0.804</td>
<td>0.757</td>
<td>&lt;0.001</td>
<td>0.650</td>
<td>0.300</td>
<td>2.362</td>
</tr>
</tbody>
</table>

For CMMI Level 5, the natural log of KNEW is the independent variable with the natural log of the actual hours in thousands as the dependent variable. The regression equation is \( \text{LN(ActHrsK)} = 1.61 + 0.662 \times \text{LN(KNEW)} \). Minitab reports no unusual observations. The graphs suggest the standardized residuals follow a normal distribution for the most part. The coefficient of determination is 0.682 with a statistically significant \( p \) value less than 0.001. The standard error is 44.657 and the CV is 0.592. MMRE is at 0.585 with PRED(25) at 0.367 – both are representative of the variation in software effort. Creating, writing, debugging, and integrating new code is key to achieving new, required functionality (Fig. 5).

The Durbin–Watson statistic of 1.47 is not as close to 2.0 as some of the other regression equations. This indicates the error terms may be independent or may slightly have positive autocorrelation.

For CMMI Level 5, the natural log of KSLOC is the independent variable with the natural log of the actual hours in thousands as the dependent variable. The points at each of the distribution tails (at the “Normality Probability Plot” upper chart) are far outside the lines. They fall outside the \([-2, 2]\) bound on the standardized residual plot, although they are in the \([-3, 3]\) bound. The histogram gaps at both ends, though the majority of the distribution aggregates in the center. Minitab reports a possible curvature in the independent variable, LN(KSLOC) with a significant overall lack of fit test (Fig. 6).

The regression equation is \( \text{LN(ActHrsK)} = 1.14 + 0.579 \times \text{LN(KSLOC)} \). The coefficient of determination is 0.757 with a \( p \) value less than 0.001 and a standard error of 60.73 with a CV of 0.804. MMRE is at 0.65, not close to the desired 0.25 or less, and PRED(25) is at 0.30, also not close to the desired 0.75 or more. The equation with LN(KNEW) performs better than this one with LN(KSLOC). However, the equation for LN(KSLOC) is useful for comparison to published literature (Jensen et al., 2006). KNEW and KSLOC represent the two bounds for KESLOC.

Table 1 summarizes the standard error, the coefficient of variation (CV), the coefficient of determination (\( R^2 \)), the statistical significance (\( p \) value), MMRE, PRED(25), and the Durbin–Watson value. The paper focuses on OLS with \( R^2 \) greater than 0.60.

6. Limitations

The small data sets threaten internal validity. Data was taken from DoD databases with paired initial and final records and from the highest CMMI level. The DoD data set has varied amounts of data from contractors with varied CMMI levels and other CER equations use only final reported data parameters. “External validity threats arise when experimenters draw incorrect inferences from the sample data” (Creswell, 2009). Future records in the SRDR repository may or may not be similar to these records.

Construct validity is manifest in “inadequate definitions and measures of variables” (Creswell, 2009). A definition of new code occurred in most, but not all, of the SRDR data dictionaries. Modification levels of 25%, 30%, and 50% were common as demarcations when to count modified code as new code. With lines of code, the type of line...
Fig. 5. CMMI Level 5 standardized residual plots for independent variable, natural logarithm of estimated $K_{NEW}$, and dependent variable (natural logarithm of actual effort hours [$K$]).

Fig. 6. CMMI Level 5 standardized residual plots for independent variable, natural logarithm of estimated $K_{SLOC}$, and dependent variable (natural logarithm of actual effort hours [$K$]).
could be logical, physical, non-commented source, and others. SLOC conversions to logical statements employed translation tables from 2010. As data and knowledge increase over time, these translation tables may become outdated, although they are nearly identical to those published in 2014 (Rosa et al., 2014). Measures of software size threaten construct validity more than the measures for any of the other variables. While there is information in the repository on changes in scope and functionality from the initial project estimate to the final estimate, these changes are only reflected in the hour count, with subsequent impact to the CER results.

7. Conclusions

After creating and testing CER equations using initial, expected parameters to predict actual, reported final effort hours, normalizing between physical and logical line counts by languages suggested by Lum et al. (2008), the different proposed CER equations for CMMI Level 5 developers used only one of the initially estimated parameters to predict final effort hours. The use of these single initially estimated parameters by the software developer obviated any need to add growth or shrinkage to the resulting CER equation for final effort hours. Data for lower CMMI level developers was too variable or too sparse to analyze. In the case of CMMI Level 4, diagnostics reported either large leverage and large standardized residuals or overall lack of fit.

Software size represented by new or total logical source statements, which are the lower and upper bounds of the effective lines of code known as ESLOC, or KESLOC, relate statistically to final effort. Notwithstanding size, peak staff estimated by the development team at the beginning of the project can be used alone to predict software development effort. These CERs can be used to provide a range of possible effort outcomes, assuming available input parameters, for CMMI Level 5 developers.

As this data set expands, other DoD researchers can use these results as a baseline to study how relationships among the variables change affecting CER equations. Analyses of high CMMI levels’ software parameters and metrics will expand knowledge of their impact on the production of software and of the impact of software environments. As these data sets grow and change, some parameters need to stay constant for future comparisons. Creating and using standard measurements of software parameters could alleviate threats to construct validity in the future. Characteristics of future data sets will need to be compared to the characteristics described.

8. Future research

As the data on software developers achieving top CMMI levels expand, continued study of CER equations by CMMI level may become common in software cost estimation. As there are two representations for CMMI (staged and continuous), it may be useful to investigate whether the representation type makes any difference in the CER equations. Repeating this method with the same data not separated by CMMI level would create CER equations to compare to those in this research. Repeating this method with similar or different data sets by process maturity level may validate the utility of CMMI level partitioning. Either by process maturity level or not, different application area foci have been explored recently (Rosa et al., 2014). Furthering the use of application areas or environments such as ground, air, and space may provide useful CER equations for future software cost estimation as the data set expands.

Comparing each of the initially estimated parameters to final parameters may shed light on over- or under-estimation, particularly by viewing them by a category of interest. For example, initially estimated peak staff may relate differently to final peak staff at various CMMI levels. As the use of COTS, GOTS, and OSS expands, measures of their usage should be recorded and examined as possible effort and cost drivers. As measures evolve, new measures appear or become standardized, particularly for software size. In these instances, new sets of CER equations should emerge. Experimenting with software development learning curves may lead to their use by one or more data set parameters. Use of similar methods with different statistical tools would reveal the impact of underlying methodologies and calculations.

Acknowledgments

Organizations providing assistance for this research were the Department of Defense and Systems Engineering and Operations Research (SEOR) Department at George Mason University (GMU). Numerous individuals provided help and encouragement. Noteworthy contributions were provided by Dr. Peggy Brouse, Dr. Andrew Sage, Dr. Alexander Levis, Mr. Justin Moul, Mr. John Moskowitz, and Dr. Wilson Rosa.

References


Corinne C. Wallshein is the Technical Advisor in the Information Technology Estimating Division and reports to the Division Director responsible to the Deputy Assistant Secretary of the Navy (Cost and Economics) to develop and defend independent cost estimates for major acquisition programs. She conducts research under the auspices of the Naval Center for Cost Analysis Technical Director with the Cost Research and Tools Division. She has a M.S. in Operations Research and a Ph.D. in Information Technology from George Mason University.

Andrew G. Loerch is an Associate Professor and the Associate Chair of the Department of Systems Engineering and Operations Research at George Mason University where he directs the track in Military Applications of Operations Research in the master’s program in Operations Research. He holds a Master of Science in Operations Research from the Naval Postgraduate School, and a Ph.D. in Operations Research from Cornell University. He is also a retired Army Colonel with 26 years of active federal service of which 15 years was spent as a military operations research analyst. Dr. Loerch is a Past President and Fellow of the Military Operations Research Society, is an associate editor of Military Operations Research, and is the editor of the book, Methods for Conducting Military Operational Analysis. He recently received the Vance R. Wanner Memorial Award for outstanding contributions to the field of Defense Analysis. He is an avid cyclist and senior softball player, the principle bassoonist of the Manassas Symphony Orchestra, and a diehard fan of the New York Yankees.