Simple Software Cost Analysis: Safe or Unsafe?

Tim Menzies*
Computer Science
Portland State University.
tim@menzies.us

Dan Port† Zhihao Chen‡
† Uni. of Hawaii, Computer
Science, Manoa;
‡ Center for Software
Engineering,
Univ of Southern California
dport@hawaii.edu,
zhihaoch@cse.usc.edu

Jairus Hihn
Jet Propulsion Laboratory
Pasadena, USA
jairus.m.hihn@jpl.nasa.gov

ABSTRACT
Delta estimation uses changes to old projects to estimate new projects. Delta estimation assumes that new costs can be extrapolated from old projects. In this study, we show that in certain real-world data sets, there exists attributes where this assumption does not hold. We define here an automatic method to find which attributes can be safely used for delta estimation.

Categories and Subject Descriptors
D.2.9 [Software Engineering]: Time Estimation; K.6.3 [Software Management]: Software Process

General Terms
Algorithms, Measurement, Economics, Experimentation, Theory, Verification

Keywords
COCOMO, cost estimation, delta estimation, stability, M5, LSR

1. INTRODUCTION
In Safe and Simple Software Cost Analysis, Boehm presents a software effort estimation method where new projects are costed via their delta to previous projects [2]. The method is simple, fast, and best of all, can take full advantage of local costing information.

A premise of that delta estimation method is that the cost of new projects can be safely extrapolated from old projects. This paper examines this safety premise using two data sets from the PROMISE repository [8]: the 1981 Cocomo81 data set used to derive the COCOMO-I model [1] and 60 NASA projects from described in CocomoNASA. All these projects come from the 1980s and 1990s.

In this study, we processed these two sets in two separate runs, as follows. In each run, we randomly sampled 3/4ths the data to learn linear models. After 30 repeats of this random sub-sampling, some attributes were found to be stable; i.e. had a similar influence on effort in all sub-samples. We say that delta estimation is safe when applied to these stable attributes.

However, many more attributes were unstable; i.e. had a very different influence on effort in the sub-samples. We say that delta estimation is unsafe when applied to unstable attributes that have a widely variant effect on effort.

One possibility is that instability is just an artifact of our sub-sampling policy degrading the performance of the learned models. To check that possibility, we carefully benchmarked the performance of our sub-samples when tested on the 1/3rd of the data not used to learn the model. These models performed as well as known high water marks from the COCOMO literature; i.e. the sub-sampling policy used here does not artificially degrade the learned models.

Our results also suggest why Boehm had not seen this instability effect before. COCOMO-I’s attributes are much stabler than the NASA projects. Hence, delta estimation would be less safe for projects from the NASA data than from the Cocomo81 data. We conclude, therefore, that before conducting delta estimation, it is important to first check for attribute instability.

The rest of this paper describes COCOMO, delta estimation, our benchmark studies, and methods for checking for attribute instability. Note that this entire approach is automatic and can quickly be applied to new domains.

1.1 Digressions
Before beginning, we offer two digressions. Firstly, our original intent was to perform this study on numerous data sets from numerous industrial sites (ideally from sites that have produced multiple versions of the same project). This proved to be an impossible goal: modern corporations are very reluctant to disclose their costing data since such data often compromises their ability to effectively compete in the

See http://menzies.us/pdf/05safewhen.pdf for a draft of this paper.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage, and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

PROMISE '05 May 15, 2005 St. Louis, Missouri USA
Copyright 2005 ACM 1-59593-125-2/05/0005 ...$5.00.
market place. The best we can do here is to base our study on the available COCOMO data. In the future, we hope to repeat this study on COCOMO-II. However, as far as we know, there is no large publicly available data base of COCOMO-II projects. Hence, to ensure reproducibility, this paper uses COCOMO-I data.

Secondly, we also intended to present summaries of our results compared to other researchers. This turned out not to be possible since, with the exception of our other PROMISE paper [4], we are unaware of research reporting variances in cost models as the the examples used in training change and the attributes in the training set also change. As to our other PROMISE paper, that work studied how feature subset selection can improve cost estimation predictive accuracy by intelligently selecting fewer attributes for the training set. However, that study does not study the stability of the attributes in the learned models (but such a stability study would be the logical next step).

2. COCOMO AND DELTA ESTIMATION

The COCOMO effort estimation model [1,3] measures effort in calendar months of 152 hours (and includes development and management hours). COCOMO assumes that the effort grows more than linearly on software size; i.e. months $\propto KSLOC^b$ where KSLOC is estimated directly or computed from a function point analysis. More specifically:

$$\text{months} = a \cdot (KSLOC^b) \cdot \left( \prod_j EM_j \right)$$

(1)

where $EM_j$ is one of a set of effort multipliers shown in Figure 1. In COCOMO I [1], the exponent on KSLOC was a single value ranging from 1.05 to 1.2. In COCOMO II [3], the exponent $b$ of Equation 1 was divided into a constant, plus the sum of five scale factors which modeled issues such as “have we built this kind of system before?”.

The numeric values of the effort multipliers are shown in Figure 2. These were learned by Boehm after a regression analysis of the projects in the COCOMO I data set, followed by extensive DELPHI sessions with estimation experts [1]. Figure 1 and Figure 2 are sorted by $EM_{max}$ where $EM_{max}$ and $EM_{min}$ are the largest and smallest effort multiplier values seen for $EM_i$. With that sort order, the effort multipliers that most reduce effort are shown at the top while the lower entries most increase effort. Note that, in COCOMO I, $seed$ has a U-shaped correlation to effort; i.e. giving programmers either too much or too little time to develop a system can be detrimental.

COCOMO can be used for delta estimation as follows. An old project with known costs is used as a baseline. The new project to be estimated is described in terms of its deltas to the effort multipliers. The new estimate is then the product of the baseline times the effort multiplier deltas. For example, suppose the new project is estimated to have the same size as the old project. In this case, we don’t need to scale the new project’s cost by a factor $\left( \frac{KSLOC(\text{new})}{KSLOC(\text{old})} \right)^b$. However, we may need to make adjustments for other changes to the project. For example, reducing analyst capability from very high to high changes the new project’s cost by a factor $0.71 = 1.21 - 21\%$ more than the old project [2].

Note that this delta estimation approach assumes that the only factors that change between projects are the factors modeled in the COCOMO parameters. While the COCOMO parameters capture a wide range of issues, they are hardly complete. This is an issue well understood from the COCOMO team who continually refine and extend the COCOMO model (for example, the COCOMO II [3] model contains several more parameters than those shown in Figure 1). In the limit, this is an issue that can never be resolved: every model contains only a finite number of attributes and so may not contain an important and relevant factor. All we can offer here is this analysis is a general method of finding stable parameters within a model.

3. EVALUATION CRITERIA

3.1 Stability Criteria

Data miners can convert COCOMO-I data into linear models of the form:

$$X = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots$$

(2)

The M5 and LSR linear model learners used in this study (from the WEKA package [9]) performs M5-style parameter pruning [7]; i.e. they step through all the attributes removing the one with the smallest standardized coefficient until no improvement is observed in the estimate of the model error given by the Akaike information criterion. Hence, some attributes may be absent from the learned model.

Under the condition of $N$ repeat sub-samples, we say that $X_i$ is unstable if (a) it is pruned away in the majority of
sub-samples (i.e. \( > 0.5 \times N \)) or (b) there is a large variance (defined below) in its associated \( \beta_i \) value. For example, suppose we learn these three models from \( N = 3 \) sub-samples of some data:

\[
\hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2 + \hat{\beta}_3 X_3 + \hat{\beta}_4 X_4
\]

**sub sample 1:**  
\[
22 + 101 X_1 + 21 X_2 + 31 X_3 + 41 X_4
\]

**sub sample 2:**  
\[
25 + 11 X_1 + + 30 X_3 + 42 X_4
\]

**sub sample 3:**  
\[
24 + 1 X_1 + + 32 X_3
\]

\[
(iii)
\]

Here, \( X_1 \) is unstable since \( \hat{\beta}_1 \) has such a wide range. Further, \( X_2 \) is also unstable because it is usually pruned away. This example was based on \( N = 3 \) repeats and this is insufficient to make reliable conclusions. Appealing to the central limit theorem, the results presented below are based on \( N = 30 \) repeats.

In the sequel, we will use the following definition of “large variance”. Given the success of the COCOMO-I model, we say that a “large variance” is one that is larger than the variances seen in COCOMO-I. This definition needs to be refined and we are working on a better definition. However, even with this approximate definition, we can show below the main result of this paper: that some attributes are very unstable indeed are hence not suitable for delta estimation.

An alternative approach to the above was proposed by one of the reviewers of this paper. Rather that check for the presence of an attribute, or the variance of its coefficient, in the learned theory another method might be to compare the variance of each independent variable with the class variable over the whole data sets. While much simpler that the study reported here, this alternate method would not study the effect of conjunctions of attributes on the target variable. Since our approach assesses the stability of attributes in the context of a conjunctive model, we have some confidence that our study catches “pair” effects where some set of attributes might be unstable in isolation but stable in conjunction.

### 3.2 Performance Criteria

A good software effort estimator generates predictions “close to” actual known efforts. One such measure of “closeness” is \( \text{PRED}(N) \) which is computed from the relative error, or \( RE \), which is the relative size of the difference between the actual and estimated value; i.e. \( RE_i = \frac{\text{estimate}_i - \text{actual}_i}{\text{actual}_i} \).

Given a test set of size \( X \), then the mean magnitude of the relative error, or \( \text{MMRE} \), is the average percentage of the absolute values of the relative errors; i.e. \( \text{MMRE}_i = \text{abs}(\text{RE}_i) \) and:

\[
\text{MMRE}_i = \frac{100}{X} \sum_{i}^{X} \text{RE}_i
\]

\( \text{PRED}(N) \) reports the average percentage of estimates that were within \( N \)% of the actual values:

\[
\text{PRED}(N) = \frac{100}{X} \sum_{i}^{X} \left\{ \begin{array}{ll} 1 & \text{if } \text{RE}_i \leq \frac{N}{100} \\ 0 & \text{otherwise} \end{array} \right.
\]

For example, \( \text{PRED}(30) = 50\% \) means that half the estimates are within 30% of the actual.

This paper will compare its results with the \( \text{PRED}(30) \) values reported in the COCOMO-I and COCOMO-II studies. Baseline values for \( \text{PRED}(N) \) from COCOMO are \( \text{PRED}(30) = 52\% \) (COCOMO-I [2, 163]) and \( \text{PRED}(30) = 69\% \) (COCOMO-II [5]). \( \text{PRED}(30) \) was selected instead of, say, \( \text{PRED}(30) \) since the most comprehensive experiment reported in the COCOMO community comes from Chulani et al. [5] who reported the results of their 15-way hold out experiment in terms of \( \text{PRED}(30) \). In such a hold-out, some fraction of the available data is selected to be a test set while the remaining data is used for testing.

Having decided to use \( \text{PRED}(30) \) from hold-out experiments, we have a problem reporting early COCOMO-I results. COCOMO-II has been studied far more rigorously than COCOMO-I and we are unaware of results from hold-out studies on COCOMO-I. The best we can say is that the COCOMO-I results are perhaps slightly inflated since they were not generated using hold-out studies (i.e. where the training set was kept separate to the test set).

### 4. DATA AND LEARNERS

This study applied LSR and M5 (two data miners from the WEKA toolkit [9]) to the 63 project instances in Cocomo81 and the 60 project instances in Cocomo-NASA. Both data sets use the COCOMO-I attributes. LSR builds one linear model through the training data while M5 can build one or more models. Internally, M5 builds a decision tree whose leaves are linear models which apply to different zones of the parameter space. Hence, M5 outperforms LSR when the data can’t be modeled as a single linear model.

Nevertheless, LSR can still model non-linear data if that data is first linearized. For example, a log transform replaces all numerics with their logarithm. Data from an exponential distribution forms a straight line in a log transformed space and hence could be modeled by LSR.

For example, Equation 1 hypothesizes that effort is exponential on program size. Figure 3 supports that hypothesis and shows a log transform on effort and lines of code in our two COCOMO-I data sets. As predicted by Equation 1, the relationship between these two variables is linear in the log transformed space.

Before we can explore stability and sub-sampling, we first must certify that our sampling and learning methods do not artificially degrade performance when learning cost estimation models from COCOMO data. Therefore, prior to the sub-sampling study, we certified our learners by trying various different learning strategies. Hence, we treated our data in several ways. Firstly, both were converted to the same set of symbols to generate cocomo81 and nasa60. These datasets looked like this:

\[
\{ xrel \ , \ data \ , \ . . . \ , \ sced \ , \ loc \ , \ effort \ \} \\
\{ nominal \ , \ high \ , \ . . . \ , \ low \ , \ 70 \ , \ 278 \} \\
\{ very \_ high \ , \ high \ , \ . . . \ , \ low \ , \ 227 \ , \ 1181 \} 
\]
Next, for $X \in \{cocl81, nasa60\}$, the following treatments were applied:

- **X_num**: The symbols very low, low, nominal, etc were replaced with 0.8, 0.9, 1, etc. Under these substitutions, nominals effect the COCOMO-I output of Equation 1 by a factor of one.
- **X_em**: The attribute values were replaced by their associated effort multipliers from Figure 2.
- **X_num_loc, X_em_loc**: All the effort multiplier attributes were removed, leaving just lines of code and effort.
- **X_num_In, X_em_In, X_num_loc_In, X_em_loc_In**: A log transform was applied to the above data sets.

To certify the competency of our method, we benchmarked our results with prior landmark COCOMO results [5]. Using the same methodology as those prior results, we ran M5 and LSR over the treated data sets using 30 repeats of a \( \frac{3}{2} \times \frac{1}{2} \) hold-out experiment. The results were converted to win/loss tables as follows. For each of the 30 selections, the training and test sets were transformed them into \( X_{num}, X_{em}, \) etc; then run through M5 and LSR. This generated a table of 30 results, with separate columns \( 1 \ldots C \) for each combination of \( \{M5, LSR\} \times X_{num}, X_{em}, \) etc. For each pair of columns \( i, j \in \{1 \ldots C\}, j > i \), a two-tailed t-test was performed to check if the mean of one column was significantly different to the other. If not, then a “tie” was declared. Otherwise, the means were compared numerically to declare a “win” or a “loss”. The resulting “win”/“loss”/“tie” counts were then sorted by “win-losses”.

The “win-loss” tables let us select the combination of learner (M5 or LSR) and the treatment (\( X_{num}, X_{em}, \) etc) that performs the best. This best combination was then used in the sub-sampling study where, 30 times, we generated \( \frac{3}{2} \) rd/train/test sets; then collected the PRED(30) scores seen when the models learned from the training set were applied to the test set.

5. RESULTS

This certification phase of this work required 960 runs:

- 2 learners + 2 data sets + 8 treatments + 30 repeats

Each run generated an effect estimator. For example, here is one effect estimator learned from nasa60_em_In by LSR:

$$\text{act\_effort} = -0.8474 * \text{rel} + 2.7259 * \text{data} + 2.9451 * \text{time} + 0.0264 * \text{stor} + 0.882 * \text{turn} + 2.8169 * \text{acap} - 6.6709 * \text{lexp} + 3.4748 * \text{lexp} - 1.8933 * \text{modp} + 1.1274 * \text{loc} + 1.4098$$

Figure 4 shows a summary of the certification results. The mean PREDs for coc81 were always lower than the mean PREDs from nasa60 (on average, by about 20%). Figure 5 shows that the variance in the estimates can be quite large: in 30 repeated trials on M5 on nasa60, the max and min PRED(30)s seen in out treated data sets can be as large as 45%. Hence, in order to compare our means, we turn to the t-test results shown in Figure 6 and Figure 7. In terms of “wins-losses”, the pattern of results is the same for both coc81 and nasa60:

- Linearization always beat non-linearization
- PREDs found using just lines of code were always worse than using all the effort estimators.
- The best performance was seen using linearized data that included the effort multiplier attributes.

In the top-performing cases, in terms of “wins-losses”:

- Using simple numbers like 0.8, 0.9, 1, etc did as well as using the effort multipliers of Figure 2.
- Neither M5 or LSR was a clear winner.

From the win-loss tables, we see that the best combination of learner and treatments is either (M5 or LSR) and \( \text{em\_ln} \) that performs the best. This best combination was then used in the sub-sampling study where, 30 times, we generated \( \frac{3}{2} \) rd/train/test sets; then collected the PRED(30) scores seen when the models learned from the training set were applied to the test set.

Figure 4 displays a Equation 3-style analysis of the \( \beta \) values seen in the 30 hold-outs with LSR on \( \text{num\_In} \) and \( \text{em\_In} \). Note that these figures only show effort multipliers that were found in the majority of the 30 hold-outs; i.e. if an effort multiplier variable appears less than 16 times, it is not shown.

The important features of Figure 8 and Figure 9 are:

- LOC is always present, with low \( \beta \), deviation.
Figure 6: T-tests on \textit{coc81}: 99% level

<table>
<thead>
<tr>
<th>learner treatment</th>
<th>Wins - Loses</th>
<th>Wins</th>
<th>Lose</th>
<th>Tie</th>
</tr>
</thead>
<tbody>
<tr>
<td>m5_num_ln</td>
<td>12</td>
<td>12</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>m5_em_ln</td>
<td>12</td>
<td>12</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>lsr_num_ln</td>
<td>12</td>
<td>12</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>m5_num_loc_ln</td>
<td>3</td>
<td>7</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>m5_em_loc_ln</td>
<td>3</td>
<td>7</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>lsr_num_loc_ln</td>
<td>3</td>
<td>7</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>m5_em</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>m5_num_loc</td>
<td>-7</td>
<td>1</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>m5_em_loc</td>
<td>-7</td>
<td>1</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>lsr_num_loc</td>
<td>-8</td>
<td>1</td>
<td>9</td>
<td>5</td>
</tr>
<tr>
<td>lsr_em_loc</td>
<td>-8</td>
<td>1</td>
<td>9</td>
<td>5</td>
</tr>
<tr>
<td>lsr_num</td>
<td>-9</td>
<td>0</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>lsr_em</td>
<td>-14</td>
<td>0</td>
<td>14</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 7: T-tests on \textit{nasa60}: 99% level

<table>
<thead>
<tr>
<th>\textit{coc81}</th>
<th>n</th>
<th>mean</th>
<th>sd</th>
<th>\textit{nasa60}</th>
<th>n</th>
<th>mean</th>
<th>sd</th>
</tr>
</thead>
<tbody>
<tr>
<td>cxpx</td>
<td>24</td>
<td>1.4</td>
<td>0.3</td>
<td>stor</td>
<td>22</td>
<td>-0.8</td>
<td>0.8</td>
</tr>
<tr>
<td>time</td>
<td>16</td>
<td>1.9</td>
<td>0.4</td>
<td>time</td>
<td>29</td>
<td>2.5</td>
<td>0.8</td>
</tr>
<tr>
<td>pcap</td>
<td>29</td>
<td>1.7</td>
<td>0.5</td>
<td>cxpx</td>
<td>16</td>
<td>1.7</td>
<td>1.0</td>
</tr>
<tr>
<td>acap</td>
<td>29</td>
<td>2.1</td>
<td>0.5</td>
<td>acap</td>
<td>28</td>
<td>2.7</td>
<td>1.0</td>
</tr>
<tr>
<td>rely</td>
<td>30</td>
<td>2.0</td>
<td>0.6</td>
<td>data</td>
<td>26</td>
<td>3.1</td>
<td>1.2</td>
</tr>
<tr>
<td>seed</td>
<td>25</td>
<td>2.9</td>
<td>0.7</td>
<td>turn</td>
<td>17</td>
<td>1.8</td>
<td>1.5</td>
</tr>
<tr>
<td>vexp</td>
<td>24</td>
<td>3.0</td>
<td>1.2</td>
<td>vexp</td>
<td>26</td>
<td>-5.6</td>
<td>2.3</td>
</tr>
<tr>
<td>loc</td>
<td>30</td>
<td>1.2</td>
<td>0.1</td>
<td>loc</td>
<td>30</td>
<td>1.1</td>
<td>0.9</td>
</tr>
</tbody>
</table>

Figure 8: LSR $\beta$ values from *em\_ln data.

<table>
<thead>
<tr>
<th>\textit{coc81}</th>
<th>n</th>
<th>mean</th>
<th>sd</th>
<th>\textit{nasa60}</th>
<th>n</th>
<th>mean</th>
<th>sd</th>
</tr>
</thead>
<tbody>
<tr>
<td>cxpx</td>
<td>24</td>
<td>2.0</td>
<td>0.5</td>
<td>data</td>
<td>22</td>
<td>1.8</td>
<td>1.0</td>
</tr>
<tr>
<td>seed</td>
<td>29</td>
<td>-3.0</td>
<td>0.6</td>
<td>time</td>
<td>28</td>
<td>4.4</td>
<td>1.2</td>
</tr>
<tr>
<td>stor</td>
<td>22</td>
<td>2.5</td>
<td>0.6</td>
<td>modp</td>
<td>25</td>
<td>2.6</td>
<td>1.4</td>
</tr>
<tr>
<td>pcap</td>
<td>28</td>
<td>-3.2</td>
<td>0.7</td>
<td>stor</td>
<td>24</td>
<td>-1.2</td>
<td>1.4</td>
</tr>
<tr>
<td>rely</td>
<td>30</td>
<td>3.3</td>
<td>0.8</td>
<td>acap</td>
<td>27</td>
<td>-5.2</td>
<td>1.7</td>
</tr>
<tr>
<td>vexp</td>
<td>30</td>
<td>-3.4</td>
<td>0.8</td>
<td>seed</td>
<td>25</td>
<td>2.5</td>
<td>2.6</td>
</tr>
<tr>
<td>acap</td>
<td>30</td>
<td>-3.8</td>
<td>1.0</td>
<td>turn</td>
<td>16</td>
<td>1.6</td>
<td>2.9</td>
</tr>
</tbody>
</table>

Figure 9: LSR $\beta$, from *num\_ln data.

- Nearly half the attributes defined in COCOMO-I, are missing (i.e. are unstable) in both data sets.
- The list of missing attributes varies from experiment to experiment; e.g. pcap appears in the majority of the \textit{coc81} results but not at all in results from the \textit{nasa60} data sets.

Also, the \textit{coc81} attributes are stabler than the \textit{nasa60} attributes:

- Most of the \textit{coc81} $\beta$, standard deviations are less than 1.0 while most of the \textit{nasa60} $\beta$, standard deviations are greater than 1.0 (and, in five cases, greater than 2.0).
- The stable \textit{coc81} attributes are well-behaved in the sense that their $\beta$, standard deviation only increases when the mean $\beta$, values increases.
- The stable \textit{nasa60} attributes are not so well-behaved. Large $\beta$, standard deviations can be found even when the $\beta$, mean is quite small; e.g. turn has one $\beta$, mean of 1.6 but a $\beta$, standard deviations of 2.9(!).

6. DISCUSSION

We set out with the intent of identifying stable attributes that can be safely used in delta estimation. Along the way, we needed to define some baselines for COCOMO-style effort estimation. Consequently, our results comment both on COCOMO effort estimation and delta estimation.

**Adequacy of our tools:** If we could not have reproduced known baselines from the COCOMO research, then our results would have been questionable. The tools described above used COCOMO-I data and LSR to achieve comparable results to known benchmarks for COCOMO-I of PRE(30)=52% (see Figure 4)\(^2\).

**Merits of just using LOC:** Figure 3 suggests that lines of code might be enough information to perform adequate software cost estimation. Figure 6 and Figure 7 shows that this is not necessarily the case. In terms of "wins-loses", estimates based on just lines of code always performed worse than linearized data sets containing COCOMO-I effort multipler attributes.

**Merits of linearization:** Equation 1 predicts that effort is exponential on program size. Our results supported this core premise of the COCOMO research. In Figure 6 and Figure 7, linearized data sets always performed non-linearized data sets. Hence, this study endorses the core COCOMO assumption that a single continuous exponential function is adequate for modeling software effort.

**Merits of multi-segmented models:** This study did not find any value is using a combination of multiple continuous exponential functions to model software effort. Recall that in Figure 6 and Figure 7, M5 never out-performed LSR on the linearized data. That is, a multi-segmented model (in log transformed space) did not do better than a model with a single-segment.

**Merits of stratification:** \textit{nasa60} comes just from the aerospace domain while \textit{coc81} comes from many domains including finance, engineering, etc. The stratification hypothesis [3] claims that specialized subset (like \textit{nasa60}) should generate higher PREs than general data sets (like \textit{coc81}). Figure 4 lends support to the stratification hypothesis since the PRE(30) means from the stratified \textit{nasa60} data sets are all 20% (approx) higher than the PRE(30) means. However, curiously, this higher PRE comes with a cost: more instability in the $\beta$, values. In the future we plan to

\(^2\)To be precise, we used hold-out studies and the COCOMO-I benchmark study did not. Also, our exact best results were a mean PRE(30) of 44.3% with a standard deviation of 10.8; i.e. the COCOMO-I results were within 70% of one standard deviation to our results.
explore this strange disconnect where instability does not mean lower PREDs.

**Merits of effort multiplier attributes:** In Figure 6 and Figure 7, the data sets using all the effort multiplier attributes defined in Figure 1 always improved PRED over just using LOC. Hence, using some effort multiplier attributes is useful. However, using all of them may not be. Figure 8 and Figure 9 showed that it is possible to ignore some of the effort multiplier attributes are still reach the adequate performance levels seen in Figure 4. Strangely, the ignorable attributes are not fixed (recall in Figure 8 and Figure 9 how different stable attributes appeared in the different tables). Hence, it is still necessary to collect all the COCOMO attributes even if they are not all used later on.

**Applying COCOMO in novel domains:** In Figure 6 and Figure 7, the data sets using the Figure 2 values did not perform better than data sets just using simpler values such as very low=0.8, low=0.9, nominal=1, etc. This means that an adequate first-pass approximation for development effort can be computed without requiring Figure 2-like values calibrated from numerous past projects. This is an important result since it means that COCOMO can be applied to totally novel domains (e.g. cost models for autonomous deep-space robots) without needing a historical record of past, similar projects (which, for novel domains, may be non-existent).

**Detecting stable attributes:** The methods used above can automatically detect stable attributes, That check means generating Figure 4 to Figure 9. Given our current tools, this check takes around 30 minutes, on a standard LINUX machine. If we adopt the coc81 results as a baseline for stability, then could declare an attribute unstable if the associated $\beta_i$ standard deviation is greater than 1.2; i.e. the maximum COCOMO-I $\beta_i$ standard deviation seen anywhere in Figure 8 or Figure 9.

**When delta estimation is safe:** Delta estimation is safe when the extrapolation from old to new projects is based on changes to stable attributes. In the case of Figure 8, there six stable attributes with a $\beta_i$ standard deviation within 1.2. Hence, for that experiment, the attributes that can be safely used for delta estimation is {loc, stor, time, cplx,acap, and data}. This list of stable attributes may not repeat in other domains (recall how the set of ignorable attributes changes in all the tables of Figure 8 and Figure 9). Hence, if the reader wishes to perform delta estimation in their own domain, it is highly recommended that they first find the attributes that are stable in their own domain.

**Acknowledgments**

This research was conducted at Portland State University, the Jet Propulsion Laboratory, and the University of Hawaii sponsored, in part, by the NASA Office of Safety and Mission Assurance under the Software Assurance Research Program led by the NASA IV&V Facility. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement by the United States Government.

7. **REFERENCES**


