Scope Error Detection and Handling concerning Software Estimation Models

Salvatore Alessandro Sarcia'
Univ. of Rome Tor Vergata – DISP
Via del Politecnico, 1
00133 Rome (Italy)
+39-06-7259-7942
sarcia@disp.uniroma2.it

Victor Robert Basili
University of Maryland and
Fraunhofer Center - Maryland
College Park, MD, 20742, USA
+1-301-405-2668
basili@cs.umd.edu

Giovanni Cantone
Univ. of Rome Tor Vergata – DISP
Via del Politecnico, 1
00133 Rome (Italy)
+39-06-7259-7392
cantone@uniroma2.it

Abstract

Over the last 25+ years, the software community has been searching for the best models for estimating variables of interest (e.g., cost, defects, and fault proneness). However, little research has been done to improve the reliability of the estimates. Over the last decades, scope error and error analysis have been substantially ignored by the community. This work attempts to fill this gap in the research and enhance a common understanding within the community. Results provided in this study can eventually be used to support human judgment-based techniques and be an addition to the portfolio. The novelty of this work is that, we provide a way of detecting and handling the scope error arising from estimation models. The answer whether or not scope error will occur is a pre-condition to safe use of an estimation model. We also provide a handy procedure for dealing with outliers as to whether or not to include them in the training set for building a new version of the estimation model. The majority of the work is empirically based, applying computational intelligence techniques to some COCOMO model variations with respect to a publicly available cost estimation data set in the PROMISE repository.

1. Introduction

Over the last 25+ years, software estimation research has been searching for the best models to estimate variables of interest (e.g., cost, defects, and fault proneness) without little concern of whether the results drawn by these estimation models (EMs) could be considered safe and reliable [6], [22].

Accuracy has been the primary criterion for evaluating estimation models. Indicators as MMRE, PRED have been defined and applied for evaluation and comparison purposes [8]. Currently, the software community is still investigating whether those indicators (or statistics) can be correctly and effectively applied to model evaluations [18] or if we need some different criterion to overcome the problem that they are not invariant from the chosen error measure [11], [21]. Nevertheless, those indicators have become standard *de facto* within the software community. Some authors, however, showed that MMRE and PRED, which are based upon the Magnitude of the Relative Error (MRE), do not actually measure bias and spread of estimation models, hence they should not be used for comparison or evaluation. [14].

Kitchenham et al. [13] refer to several sources of error, i.e. errors found in the measurements (Measurement error), produced by unsuitable mathematical models (model error), wrongly assumed input values (assumption error), or inadequacy of the projects chosen for building such estimates (scope error). In this work, we refer to the latter from an uncertainty point of view. It means that, we detect and analyze estimation models by considering the probability distribution drawn by an error sample over a number of past projects (history of use). Then, we quantify the uncertainty related to the estimation model for the next project by calculating a Bayesian Prediction Interval (BPI), which is a range, calculated though Bayesian statistics, where most probably the next estimate will fall for a fixed acceptable confidence level (e.g., 90% or 95%). If the expected prediction error does not agree with the range (i.e., it falls outside the BPI), then a scope error may occur meaning that, in spite of what we expect, the expected prediction error cannot be suitably explained and additional variables and/or observations should be considered. Conversely, if the expected prediction error falls inside the range, we can argue that the scope error is acceptable and it does not affect the estimation model too much.

BPIs were defined by the authors [20] to overcome the unsolved problems left out by the traditional prediction interval (PIs). Unlike the latter, the proposed approach allows avoidance of any specific assumption about the model and provides much smaller prediction intervals. The authors showed that, BPIs quantify uncertainty more efficiently and effectively than previous methodologies [20], making the new approach more attractive and useful for evaluating model risk and uncertainty.

The problem that we deal with in this work is that, when a scope error occurs, i.e. the estimation model is built upon data that is not adequate for the project being estimated, estimates drawn by the model may have an unpredictable variability meaning that the estimation model would be too risky for correctly estimating quantities of interest. The aim of our work is to support software organizations [2] in detecting a scope error occurrence beforehand, i.e. before using the model for prediction, and consequently letting organizations make more informed decisions as to their variables.

To achieve this aim, based upon artificial neural networks, we execute a *similarity analysis* between the past projects (used for calibrating the estimation model) and the one being estimated. It means that, the scope error analysis that we suggest concerns the analysis of similarity between the project being estimated and the

historical ones. Similarity analysis aims at figuring out whether the model has been built upon a sufficient number of data points to allow estimating the new one correctly. If not, we should gather new data points or even change the model (e.g., by considering different variables or shapes) and/or the estimation technique. The scope error analysis that we first present in this paper is also a model improvement technique, which allows figuring out whether and how to increase the estimation model.

This paper is organized as follows: a discussion of related work; a description of estimation models and their evaluation with a brief explanation of Bayesian Prediction Intervals (BPIs); definition of the scope error problem; presentation of the prior scope error evaluation algorithm and the posterior scope error analysis; a case study on NASA 93 COCOMO data set [23]; and a conclusion and some hints about how to use the proposed technique in real cases.

2. Related work

The software estimation literature is huge and is mainly about cost estimation. For this reason, we will refer to cost estimation as our primary source for explanation and example. Surprisingly, however, there is little research concerning scope error, risk, and uncertainty. But there are a few interesting papers that address these topics.

Kitchenham et al. [13] refer to scope error as a risky situation to be avoided. In fact, scope error should be considered by project managers as a severe situation because, if we have never dealt with projects similar to the one being estimated, the predictive capabilities of the EM, calibrated on those projects, may be unpredictable and the risk of using the model for prediction may increase. They propose applying the portfolio concept, where the organization deals only with software systems for specific segments of the market, e.g., medium size software systems for banking environment, space agency management systems, embedded control systems. We argue that, portfolio does not actually sort out the problem. It tries to overcome the problem by escaping from the risky situation stated above. In other words, portfolio does not need to perform a *similarity analysis* because the projects included within the portfolio are assumed to be similar. When we are not able to make such an assumption, portfolio cannot be applied correctly. Moreover, all of the cross-context organizations cannot apply portfolios because their business is based upon dealing with different projects over different contexts meaning that portfolio makes no sense to those organizations.

Jørgensen et al. [12] deal with this issue by assuming that they are able to select historical projects on which the estimation model has the same accuracy. However, such an assumption might lead to selecting very different projects such that uncertainty may be incorrectly calculated. They also suggest applying *cluster analysis*. Usually, it is used in place of executing a *similarity analysis*, even though it would not be completely correct. In fact, *cluster analysis* groups projects based upon a chosen criterion such as accuracy of the estimation model. For instance, we may group together projects on which the estimation model has same accuracy. However, we argue that, once we identify project-describing characteristics we should group those projects by similarity based upon those characteristics, not based upon accuracy. Then we can investigate whether the estimation model would have the same accuracy over similar projects.

Angelis et al. [1] deal with this problem by applying the bootstrap method. Once they select similar projects to make a baseline for prediction, the selected data points may not be enough to make any prediction, i.e. calculating statistics [24]. Then, a resampling procedure such as bootstrap may somehow enlarge the data set for calculating and making significant the required statistics. Such an approach has been recently applied by Port and Korte for studying the reliability of specific accuracy indicators [19]. However, it is important to note that bootstrap cannot assure that the uncertainty is correctly calculated because its calculation is based on the assumption that the resampling procedures are relevant for the considered context, when it may not be the case.

3. Estimation models

Estimation models (EMs) that we refer to are based on parametric models represented by a function f_R such that $y = f_R(x,\beta) + \varepsilon$, where x is a set of independent variables, y is the dependent variable, and $\boldsymbol{\beta} = (\beta_0 \dots \beta_Q)$ is a set of parameters defining f_R. The component ε is the aleatory part of the model (i.e. the unknown part depending on the probability of future events) representing our uncertainty on the relationship between independent and dependent variables, with $E(\varepsilon) = 0$ and $cov(\varepsilon) = \sigma^2 I$ [24]. Usually, $f_R(x,\beta)$ is assumed linear, i.e., $f_R(x,\beta) + \varepsilon = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_Q x_Q + \varepsilon$. Parameters $\beta = (\beta_0 \dots \beta_Q)$ cannot be calculated because we do not know all the points of the population. We can only estimate β by finding a set of estimators $B = (b_0 \dots b_Q)$ such that they minimize an error function, e.g., the least squares error function. Elements of B are different from parameters β. Thus, f_R fed with the input values provides $Y_{est} = f_R(X,B)$, not Y. Then, the difference $e = Y - Y_{est}$ is a vector of errors representing ε (called residuals, with $e \neq \varepsilon$).

If we want to get the best linear unbiased estimators (BLUE) of β (Gauss-Markov theorem) [24] and use the model for inference, LS requires some regression assumptions:

- Errors ε are not x correlated
- The variance of the errors is constant (homoscedasticity), $cov(\varepsilon) = \sigma^2 I$
- Errors ε are not auto-correlated
- The probability density of the error is Gaussian, $\varepsilon \sim NID(0, \sigma^2 I)$, i.e. there are no outliers, skewed/kurtotic distributions, and measurement error.

Residuals, or some related measures (e.g. relative residuals), are used to calculate the accuracy statistics (also called accuracy indicators). It means that, we may use an Absolute Error (AE), i.e. $AE = Actual - Estimated = e = Y - Y_{est}$. However, an absolute error measure makes no sense when dealing with estimation model accuracy for software engineering variables (e.g., effort, defects). In fact, an Absolute Error would increase with the size of what we were predicting. Therefore, an absolute measure usually is not used for model comparison. When predicting software engineering variables, one of the most common choices is to consider a relative error measure, i.e. a measure that takes into account the size of what we are predicting [17], [5]. For this reason, Boehm [5] defined the performance of his software cost model (COnstructive COst MOdel, COCOMO) in terms of *Relative Error* (RE), Eqn. 1.

$$RE = (Actual - Estimated)/Actual = (Y - Y_{est})/Y. \quad (1)$$

Note that, currently Boehm's model has been enhanced into the COCOMO-II [7], but the accuracy evaluation principles have not changed. The evaluation procedure consists of taking a data set, splitting it into two parts (2/3 and 1/3), and using the 2/3 part as a training set and the 1/3 part as a test set [17]. Accuracy is evaluated by calculating two summary statistics over the Relative Error test sample. In particular, $Mean(RE_{TestSet}) = (1/TE)\Sigma_{i=1..TE}(RE_i)$ is a measure of the *bias* of f_R and $STD(RE_{TestSet}) = SQRT\{(1/N)\Sigma_{i=1..TE}([RE_i - Mean(RE_{TestSet})]^2)\}$ is a measure of the *spread* of f_R, with TE = {cardinality of the Test Set}. To evaluate scope error, we will proceed differently. We consider each data point belonging to the test set individually hence we do not calculate any statistics over the test set. In fact, we are interested in showing the procedure of detecting scope error over a sufficient number of trials represented by the test set.

Since learning organizations [2] find it much more useful to have two-point predictions instead of one-point predictions, in fact, nobody would think that one-point predictions would be the right ones [16] while a two-point prediction may give much more confidence to project managers. We argue that, to quantify uncertainty, we have to use a prediction interval, which is a two-point estimate. Mathematically, a prediction interval is defined as a range where the next estimate may fall with a stated confidence. For instance, <0.95, 120 PM, 180 PM> means that the effort in PM (Person Month) for the next estimate will be within 120 and 180 with a confidence of 95%. However, traditional approaches to calculating prediction intervals fall short of providing useful ranges because they usually are too wide to be used in real cases. For this reason, authors have defined Bayesian Prediction Intervals showing the utility of applying them in a real software development context [20].

Note that, along with estimate prediction intervals, we can also take into account error PIs. Error PIs can be obtained by PIs through an algebraic transformation, as well as estimate PIs can be calculated back from error PIs [12]. In fact, the Jørgensen and Sjøberg's approach consists of first calculating an error PI, which is a relative error range, e.g., $[RE_{DOWN}, RE_{UP}]$, and then turning it into an estimate PI. In particular, since RE = (Act – Est)/Act, then we can get Act from it, i.e., Act = Est/(1 – RE). The Estimate Prediction Interval stems from substituting $[RE_{DOWN}, RE_{UP}]$ into RE, i.e., $Act_{DOWN} = Est/(1 - RE_{DOWN})$ and $Act_{UP} = Est/(1 - RE_{UP})$. Therefore, the estimate PI is $[Act_{DOWN}, Act_{UP}]$, which corresponds to the error PI $[RE_{DOWN}, RE_{UP}]$. Note that, such an algebraic transformation can be applied not only to a Relative Error (RE), but also to a Balanced Relative Error (BRE) as originally done by Jørgensen and Sjøberg.

Because of this algebraic transformation, we can use error PIs and estimate PIs interchangeably. We use an error PI when aiming at analyzing performances of an EM from a risk point of view. We use an estimate PI when aiming at investigating an EM from a prediction point of view.

4. Bayesian prediction intervals

Bayesian Prediction Intervals (BPIs) have been defined by the authors [20] and used for defining a new way of quantifying uncertainty for model selection [21]. This new methodology of calculating BPIs can eventually be bootstrapped or included in a Markov Chain Monte Carlo simulation framework. However at this early stage, we prefer adopting neither resampling nor simulation strategies so as not to complicate the methodology. The novelty is that the authors empirically showed that using BPIs is more useful and efficient for practitioners (e.g., dealing with real cases) than exploiting traditional Prediction Intervals (PIs). This is because, BPIs provide dramatically narrower intervals than traditional ones and do not need any of the regression assumption required by PIs. As a consequence of that, BPIs can be really used for quantifying uncertainty in practical problems (where usually regression assumptions do not hold), while traditional ones have only a theoretical importance.

The proposed strategy of estimating BPIs is based on building a posterior empirical distribution of Relative Errors, Eqn. (1), and using it for inferring the uncertainty related to the estimation model with a predetermined confidence (called credibility), e.g. 95%. Therefore, the higher the extent of the BPI, the more the uncertainty of the estimation model grows. BPIs do not need any regression assumption, hence we do not assume that the distribution is a Gaussian and the sample is homoscedastic (constant variance). Instead, we consider the distribution asymmetric, affected by outliers, and with variable spread. Moreover, we let RE be x-correlated.

For the sake of completeness, we describe below the procedure for building Bayesian Prediction Intervals also reported in [20] and [21]:

(1) We calculate the non-linear robust regression function of y with respect to x, e.g. y would be RE and x would be KSLOC in a two-dimensional space. We call this regression function as EReg, which stands for Error Regression (Function). If we consider a multidimensional space the non-linear robust regression y has to be calculated with respect to a number of independent variables, e.g. $x_1, x_2, \ldots, x_N$. This kind of regression function provides an x-dependent median, minimizing the Minkowski R-distance (R = 1). If we find no correlation from the regression analysis of EReg, then we can only consider the median of the distribution of RE, omitting the independent variables $x_1, x_2, \ldots, x_N$ (i.e., considering univariate instead of multivariate samples).

(2) To deal with the heteroscedasticity issue, we estimate the x-dependent variance empirically. The strategy is based on turning the problem into a two-class discrimination problem. In particular, we use the x-dependent median (or just the median in the case of no correlation) calculated above (y) for splitting the sample into two classes; class A (the space above the x-dependent median) and class B (the space below the x-dependent median). We use observed elements of classes A and B as representatives of the unobserved data points of each class.

(3) Then, we train a Multi-layer Feed-Forward Neural Network for Discrimination called Bayesian Discrimination Function (BDF) because its output can be interpreted as the posterior probability that any input belongs to class A [4], so that the BDF output provides a measure of how far the input is above or below the x-dependent median (binomial choice). Figure 1 shows a univariate case where class A is just on the right of the median and class B is on the left.

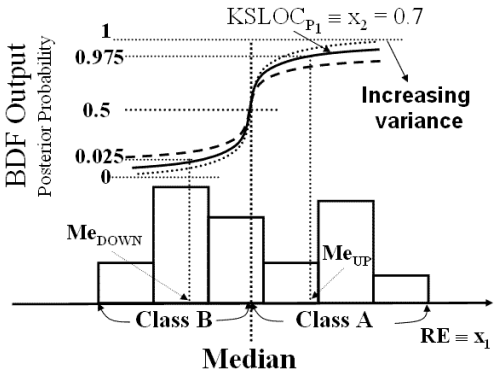

Figure 1. Posterior probability density obtained by fixing KSLOC and letting RE vary. Dotted and dashed lines represent posterior probability functions with an increasing variance.

As an example (Figure 1), a project (e.g., P_1) is classified as belonging to class A if $f_{BDF}(P_1) \geq 0.5$ and a project (e.g., P_2) is classified as belonging to class B if $f_{BDF}(P_2) < 0.5$. Actually, we are not interested in classifying our new observations. Our aim is to use the BDF for inference, i.e., fixing the posterior probability range (e.g. [0.025, 0.975]) and getting the corresponding relative error interval on the x axis, i.e. (Me_{DOWN}, Me_{UP}). Note that, the BDF performs a similarity analysis between the characteristics of the project being estimated and the observed projects upon which we built the BDF (i.e., x_1, x_2, …, x_N).

Assume now that instead of fixing both values x_1 = RE and x_2=KSLOC, we fix only x_2 (= constant c), and let x_1 vary. That is, the variable representing the relative error is not fixed while the variables representing the independent variables of the estimation model (i.e., x_1, x_2, …, x_N) are fixed. Therefore, we can now answer the question: based on the project characteristics of this project (i.e., x_1, …, x_N), what is the expected relative error range.

(4) Once we build the posterior probability density (solid line in Figure 1), we can obtain a Bayesian PI by fixing a 95% confidence level (credibility), i.e. (0.025, 0.975), and selecting the corresponding values of RE on the x-axis, i.e. (Me_{DOWN}, Me_{UP}). This interval represents the expected range where the next RE will fall, i.e., the Error Bayesian Prediction Interval. Based on the slope of the posterior probability density (Figure 1), we can evaluate the variable spread of the distribution. The slope gets steeper as the variance decreases; it gets flatter as the variance increases [10].

(5) To calculate the Estimation Bayesian Prediction Interval (e.g. over the effort) from (Me_{DOWN}, Me_{UP}), we first consider the Eqn. (1), RE = (Actual – Estimated)/Actual and then, inverting Eqn. (1), we deduce Actual = Estimated/(1 – RE).

As shown by Jørgensen et al. [12], the PI is then [Est/(1–Me_{DOWN}), Est/(1–Me_{UP})] that can be written as follows $[O_{est}^{N+1}/(1-Me_{DOWN}), O_{est}^{N+1}/(1-Me_{UP})]$, where O_{est}^{N+1} is the estimate on the next point, i.e., (N+1)th data point. For instance, assuming that the RE interval obtained from Figure 1 is [-0.9, 0.1] and the Estimated effort = 3 person months, the PI is then [3/(1-(-0.9)), 3/(1-0.1)] =[1.6, 3.4] person months.

Although the proposed prediction interval shown in Figure 1, i.e. (Me_{DOWN}, Me_{UP}), has been derived empirically without making any specific assumptions, it actually represents an underestimate of the actual uncertainty. That is because the BDF was derived from the principle of maximum likelihood estimate (MLE) that considers only the most probable parameter set. We have to consider the uncertainty over the unknown parameters of the BDF, as well. To correct this underestimation there are some techniques, though none of them is applicable in every case [15], [9]. MacKey proposes to apply the Bayesian framework. In our view, however, his approach is too computationally cumbersome and based on too many approximations and assumptions (e.g. normality). Another approach to correct the underestimation can be to apply Markov Chain Monte Carlo (MCMC) simulation, which avoids the Gaussian approximations, but is also computationally expensive and its reliability depends on the simulation assumptions.

(6) To avoid the problem of too high a computational cost without any promise of getting better results, we prefer estimating this additional uncertainty through the generalization error provided by cross-validation (leave-one-out or K-fold). This procedure is more convenient because the cross-validation procedure has to be performed anyway when selecting the classification model and it has a comparable computational cost with respect to the bootstrap or MCMC procedures. Therefore, the proposed procedure would be more convenient from a practical point of view avoiding any specific assumptions.

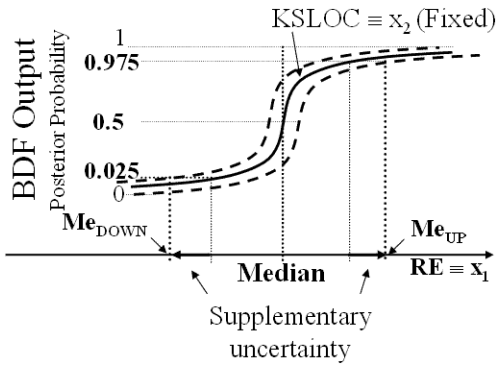

Figure 2. Posterior probability density (solid line) obtained by fixing KSLOC and letting RE vary with supplementary uncertainty due to the error in calculating the model parameters.

Our approach is based on correcting the underestimation by summing/subtracting the K-Fold CV score to the (Error) Bayesian Prediction Interval calculated in Figure 1. The two outer dashed lines in Figure 2 represent the resulting curves after summing and subtracting the additional uncertainty (generalization error associated to the BDF). In particular, the upper dashed line is $f_{|x_2=c}(x_1) + \text{CVScore}$ and the lower dashed line is $f_{|x_2=c}(x_1) - \text{CVScore}$. The upper and lower shifts determine an increase to the magnitude of the prediction interval (Me_{DOWN}, Me_{UP}) due to the supplementary uncertainty.

The final prediction interval can be derived by fixing the 95% confidence as above, i.e. (0.025, 0.975), and selecting the corresponding values of RE on the x-axis. In particular, Me_{DOWN} is calculated by the crossing point between the 0.025-horizontal line and the upper dashed line. The Me_{UP} is calculated at the crossing point between the 0.975-horizontal line and the lower dashed line.

5. Problem with scope error

We have posed the scope error problem as the situation where the EM uncertainty may be unpredictable because the project being estimated is quite different from the projects used for calibrating the EM. This situation should be considered by project managers and practitioners as a severe (risky) one because, if we have never dealt with similar projects to the one being estimated, the predictive capabilities of the EM may be unpredictable meaning that the use of the EM would be too risky.

Unlike portfolio, our approach allows an organization to take advantage of every piece of information gathered by past projects even though they are not similar to each other. Organizations need to group past data into sets of similar projects in an effective and efficient way. In particular, an effective and efficient way of doing that is to use automatic discrimination tools such as the ones dealt with in the computational intelligence field. To this end, we will mainly refer to artificial neural networks, though further techniques may be applied, as well. We call the project grouping a *similarity analysis*, i.e. we quantify the degree of similarity between the project being estimated and the past data used for calibrating the considered estimation model. Then, based on whether or not the EM has the same accuracy on the similar projects, we can detect when a scope error affects the EM. The similarity that we refer to is properly about grouping observations that are of similar type (i.e., they have close/same values for the same variables) and consequently getting the expected uncertainty from each group. We explain this concept through an example.

Table 1. Similarity Analysis – Past Observations (Act)

Similar Project Set	Effort	KSLOC	Cplx	Error PI
A	70-80	20-30	High	[-0.30;0.20]
B	50-60	60-70	Low	[-0.10;0.25]

In Table 1, there are two project sets having their own error prediction interval (Error PI). In Table 2, there are three estimated projects, i.e. they are described by estimated values. We are interested in knowing, for each project, the prediction interval where the error will fall. If our estimates are correct (e.g., estimates for KLOC, Cplx, and Effort), P1 can be classified as belonging to set A, therefore the expected error prediction interval would be the same as [-0.30; 0.20]. Project P2 has some characteristic close to set A (i.e., KSLOC = 20) and some characteristic close to set B (i.e., Cplx = 'Low'). To what extent is "close"? This uncertainty situation on project P2 is workable.
For instance, we can consider an error prediction interval as the union of the A and B prediction. The union of two distinct ordered sets is equal to or wider than each individual set. As a result, the estimation uncertainty of the estimate of project P2 grows.

Consider project P3. This represents a different situation from the previous ones. We do not have any characteristic similar to set A and B even though we may argue that KLOC = 40 is in between KLOCs of set A and B, and Cplx = "Very High" is closer to "High" than "Low". Then, estimating project P3 with the considered EM is additionally risky. That is, this situation refers to a possible scope error.

Table 2. Similarity Analysis – Project Being Estimated (Est)

Projects	Est Effort	Est KSLOC	Est Cplx	Similar Project Set	Est Error PI
P1	70	20	High	A	[-0.30;0.20]
P2	15	20	Low	A or B ?	[-0.30;0.25] ?
P3	100	40	Very High	?	?

Therefore, if we used the EM for estimating P3 we may have some unpredictable spread error. From a practical point of view, project managers and practitioners would have to recognize whether a scope error could happen and consequently evaluate risk and uncertainty as to those situations. To this end, the estimate of project P3 should be considered as more risky than the estimate of project P2 even though we may not quantify the error prediction interval for P3. Note that, the scope error does not affect the EM parameters. It is about the estimates because of the improper use of the EM. The scope error is a risky situation because we do not know the error prediction interval for similar projects even though we may state some rough limits; therefore, the risky situation arises from the fact that, we use an EM without knowing its actual uncertainty. Recognizing the scope error is not so easy when dealing with a number of variables. As an example, consider the situation where instead of having only one variable as in Table 1 (i.e., Cplx) in addition to Effort and KSLOC, we had 15 variables, as the COCOMO model, and for each of them, there were, for instance, five values. Then, there would be 5^{15} = 30,517,578,125 different subsets to be considered in addition to the ratio scale variables. Note that, many of those subsets may have the same uncertainty, so the number of distinct uncertainty intervals may be much less than 5^{15}. However, the huge number of elements shows that recognizing whether a scope error occurs, project managers and practitioners need to use evaluation systems, which are able to provide a similarity measure automatically (i.e., computationally); otherwise they will not be able to figure out the real prediction uncertainty.

6. Prior scope error evaluation algorithm

In this section, we show the way of dealing with the scope error problem before using the EM for prediction. In the next section we deal with the same problem, once we know the actual values of the project.

Based upon the methodology defined in [20] and recalled in Section 4, we build a specific artificial neural network, called Bayesian Discrimination Function (BDF), which can be inverted (because it is a smooth and monotonic), such that it provides a BPI for the project being estimated. It is worth noting that, we can perform this kind of analysis before estimating the project. That is why we define this evaluation as "prior". Then, we can evaluate whether using the EM safely, improving the EM by adding/removing some variables, changing the model shape or even changing the estimation technique. The main novelty of this technique is that, this prior analysis concerning scope error is a pre-condition to the safe use of the EM. Even though organizations do not use parametric models (e.g., human judgment-based techniques), they can still consider past error samples of the technique used for building a BDF and its inverted function. Then, based upon the inverted BDF, organizations can obtain BPIs for detecting and analyzing scope error. Our proposal is that, the prior scope error detection algorithm proposed here

could be applied as a safe pre-condition apart from the technique used for estimation.

Since this analysis is based upon the BDF, unlike traditional error PIs, error BPIs take into account any kind of model error including heteroscedasticity, x-correlation of the error, non-normality of the error distribution, non-linearity [20]. In Section 5 (Table 2), we described two different problems with scope error. The first one involved a project P2 whose characteristics had been partially observed previously in projects A and B (Table 1). The second one involved a project P3 never observed before. Situations 2 and 3 in Figure 3 show how to deal with both kinds of scope error.

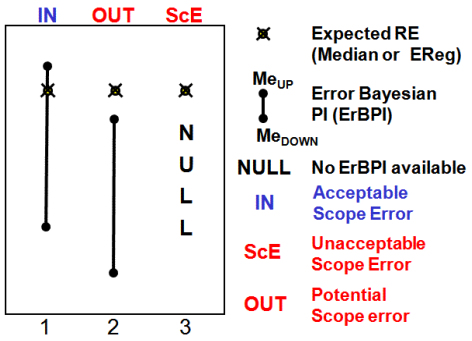

Figure 3. BDF reliability and scope error analysis.

In situation no. 1 (Figure 3), the error BPI provided by the BDF includes the expected relative error (RE), where the expected relative error is provided by the RE median or a specific regression model (EReg). In this case, the error BPIs provided by the BDF can be considered reliable and accurate hence the scope error is acceptable, i.e. it does not affect the estimate too much. In situation no. 2, the error prediction interval provided by the BDF does not include the estimated relative error. Then, it can be a warning that the BDF is unreliable and the error prediction interval is inaccurate, i.e. the scope error may be unacceptable. Actually, we are not sure whether a scope error may occur before knowing the actual values of the project. We have to check the actual values once they are available (see Section 7). Situation 2 may show that the data set has an insufficient number of observations to allow building the BDF. In that case, we would have a scope error. Conversely, the actual observations may exclude this conjecture when the actual relative error would fall within the interval. This may happen if the median or the x-dependent median (EReg) would not correctly represent the expected relative error. In that case, there would not be any scope error. For prediction purposes, however, before knowing the actual values, we should consider situation 2 in Figure 3 as a potential scope error.

Note that, situation no. 2 (Figure 3) can be improved. We can make the error prediction interval accurate even though the BDF stays biased. This can be done by increasing the (upper or lower) interval endpoint beyond the expected RE so that the interval can contain the expected RE itself. This kind of improvement is only useful if the magnitude of the final error BPI remains acceptable. If the magnitude exceeds the acceptable threshold (e.g., RE = 0.3 fixed by the Organization), this correction is not recommended. Therefore, situation no. 2 may not be a problem because it can be turned into situation no. 1 (increasing uncertainty).

With respect to situation no. 3, there is no error prediction interval to use, i.e. the BDF is not able to provide any interval. This happens when the historical data used for building the BDF did not include data points similar to the project being estimated. The Backpropagation algorithm cannot generalize information if no information is available [20]. Therefore, we can argue that a scope error will occur in situation 3. Mathematically, this case is not about "overfitting" [20]. Technically, the point here is that the classification capabilities of a neural network decrease when it is no longer able to provide significant results. For these situations, researchers and practitioners are used to designing new experiments and gathering new information. Conversely, we exploit this Backpropagation's characteristic to detect and evaluate the scope error impact on the relative error.

7. Posterior scope error analysis

Once we know the actual values of the estimates, we can figure out whether scope error affected the estimates and consequently reuse new data for building an extended version of the EM. This is the reason why, the posterior scope error analysis dealt with in this section is also referred as scope extension analysis of an EM. For a complete explanation concerning model improvement techniques, see [20].

Situations 1.a and 1.b in **Error! Not a valid bookmark self-reference.** refer to situation 1 in Figure 3, where the expected RE falls within the interval. In situation 1.a there is no scope error (i.e. no scope extension). Both the expected RE and the actual RE fall within the error BPI. Rebuilding a new instance of the estimation model including the actual project data in 1.a does not extend the scope of the EM. Therefore, situation 1.a shows that the new project is completely compatible with projects used for building the EM.

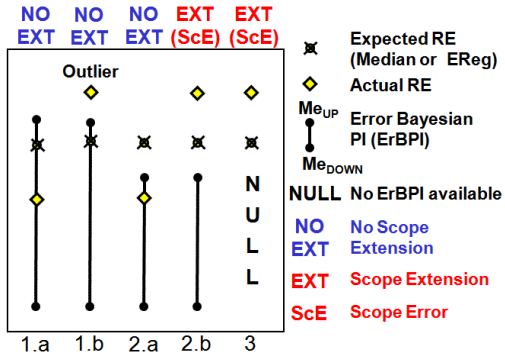

Figure 4. Scope extension analysis on Figure 3.

In situation 1.b, there is no scope error, as well. This is because the error Bayesian Prediction Interval (ErBPI) is reliable (ErBPI includes the Expected RE), but the actual RE falls outside the interval. This situation is similar to the traditional evaluation of outliers in statistics. If we included the project in 1.b in the training set when building a new version of the EM, we might observe an inclusion of the expected RE within the interval, but the model would not extend its scope. Therefore, situation 1.b would lead to increasing the uncertainty of the EM, even if it may make the error prediction interval closer to reality (less optimistic view).

Therefore, we should decide whether it is worth including that project data in the training set for building a next version of the EM. In fact, situation 1.b may imply that some kind of model

error would affect the EM in an unacceptable way. In that case, we may decide not to include the project data in the next building procedure of the EM since we would not remove the model error (e.g., finding the missing variables, the right model complexity). However, when we have a few data points to build the EM, as is usually the case in software engineering, the best we can do is to use projects like those in 1.b to build a new version of the EM.

Situations 2.a and 2.b in Figure 4 refer to situation 2 in Figure 3, where the expected RE falls outside the interval. In situation 2.a, we can make sure that no scope error occurs because the actual RE falls within the interval even though the interval does not include the expected RE. This means that the BDF was biased because of the median rather than for a lack of observations. Therefore, situation 2.a is similar to situation 1.a apart from the fact that the relative error median (or EReg) did not represent correctly the expected value of the RE.

Situation 2.b refers to situation 2 in Figure 3. Unlike situation 2.a, situation 2.b shows the actual RE falling outside the interval. This situation occurs not because the expected RE did not correctly represent the expected value, but the problem is that there are not sufficient observations to build the BDF. Therefore, situation 2.b refers to a scope error because of this lack of observations. Rebuilding a new instance of the estimation model with that data point extends the scope of the EM and we would observe a shorter distance between the actual RE and the interval, or possibly an interval that includes the RE. Note that, this kind of improvement would increase the EM uncertainty making the prediction interval more credible.

Situation 3 in Figure 4 refers to situation 3 in Figure 3. We duplicated it for the sake of completeness. As we have already explained in Figure 3, situation 3 leads to increasing the scope of the model. This happens when there is no interval because the data set did not have data similar to the considered project. Therefore, if we rebuilt the EM by including the project in the training set, we would observe a scope extension. Of course, including only one data point may not be sufficient. Nevertheless, situation 3 leads to extending the EM scope.

8. The case study

In this section, we apply the strategy presented in Sections 6 and 7 to the NASA 93 COCOMO data set [23] by considering models having different features and shapes. Of course, this data set and model are 25+ years old, so they have a marginal relevance to the field. Nevertheless, because their notoriety, they can be used effectively for showing the way of applying the proposed technique on real data.

Consider the situation where NASA is the learning organization [2] using measurement [3] and, from 1971 to 1987, they developed 8 projects, e.g., Hubble Space Telescope, involving 93 software systems. We start with our analysis at the beginning of 1985, when NASA has already developed 77 software systems so their experience is based upon those 77 software systems, which we will use as the basis for building our estimation models. The NASA's goal is to figure out whether data used for calibrating the EM is sufficient for estimating 16 next software systems (from 1985 to 1987), i.e. whether scope error may affect the estimates of those 16 projects. Since we actually know the data from these 16 software systems, we can use them as a test set for evaluating the proposed technique over 16 cases separately.

The data set [23] is composed of 93 project instances. Each instance is described by 24 attributes (Table 1). In particular, "Size", the 15 COCOMO-I multipliers, "Effort", and 7 attributes describing further characteristics of the NASA software system (project ID, project name, category of application, flight/ground system, NASA center, "YEAR" finished, and development mode). Note that, "Effort" is measured by calendar months of 152 hours, including development and management hours [5].

Based upon data from the 77 software systems, our case study consists of calibrating 2 different models comparing the variations of uncertainty intervals to each other and answering the question whether the models can be safely used for prediction. Conclusions enacted from this case study offer some new insights for the projects developed at NASA.

Table 3. Data Set Description.

Attribute	Description	Example
Size	Continuous	112.3 KSLOC
15 COCOMO-I Multipliers	Discrete (Ratio)	"very low" = 1.46
Effort	Continuous	117.2 Man-Months
ID	Categorical	1, 2, … 101
Project name	Categorical	"gal" = Galileo
Category of application	Categorical	"avionics", "science"
Flight/Ground system	Categorical	"F" or "G"
NASA center	Categorical	"center 3"
Development mode	Categorical	"embedded"
YEAR	Discrete (Interval)	1986

Models that we consider are the following:

- *LogLin*: log-linear regression function, i.e. Effort = *LogLin*(Size, 15 COCOMO-I Variables);
- *LogLinMode*: log-linear regression function with the categorical variable Mode, i.e. Effort = *LogLinMode*(Size, 15 COCOMO-I Variables, Mode).

We show two different models (*LogLin* and *LogLinMode*) to provide a variety of results and consolidate the execution of the procedure for anyone interested in duplicating this case study or applying the proposed technique over their own data.

To code categorical values, we use dummy (dichotomous) variables. If we have C categorical values, we create (C – 1) dichotomous variable, i.e. each variable has two values {0 or 1}. Since "Mode" has three categories ("Semidetached", "Embedded", "Organic"), we included two dummy variables (D1, D2) in *LogLinMode* and coded the categories as follows, "Semidetached" became {D1 = 0, D2 = 1}, "Embedded" became {D1 = 1, D2 = 0}, "Organic" became {D1 = 0, D2 = 0}. Therefore, the categorical variable Mode was represented by a couple of dummy variables.

We only used the attribute "YEAR" to split up projects into training and test sets, i.e. training set (before 1985) was composed of 77 software systems, and test set (after 1984) was composed of 16 software systems.

To focus the goal of our study, we defined a **GQM template** [3]:

Analyze the scope error of the regression models *for the purpose of* detection and improvement *with respect to* the magnitude of the Error Bayesian Prediction Intervals from the *point of view* of the project managers in the *context of* NASA's projects.

9. Results

To detect a scope error from the two considered estimation models, we first calculated the BDF for each model exploiting data from 77 projects belonging to the training set. Then, we fed the test set (remaining 16 software systems) into each inverted BDF and obtained 16 Error Bayesian Prediction Intervals (ErBPIs) for each estimation model (Figure 5 and Figure 6). Since the NASA 93 COCOMO data set only includes actual data, prior analysis shown in Figure 3 is not applicable, i.e., Bayesian PIs obtained by running both prior and posterior analysis provide the same ranges. For this reason, we only show the posterior analysis, which includes actual relative error values, as well.

With respect to Figure 5 and Figure 6, the y-axis represents the relative error (RE) and the x-axis contains the 16 software systems belonging to the test set. ErBPIs are represented as vertical segments. Crossed circles represent the expected estimation relative error (median) for the test set.

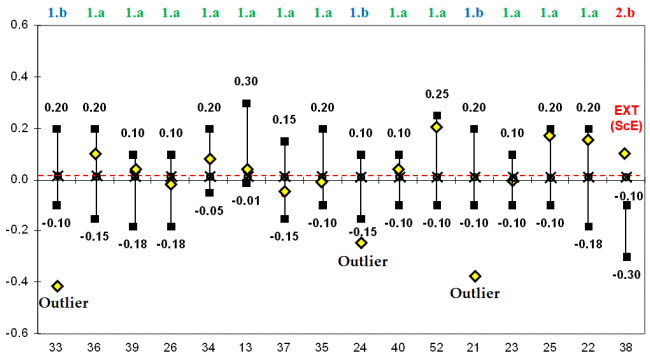

Figure 5. Error BPIs concerning *LogLin*.

In Figure 5 (*LogLin* model), 12 cases out of 16 (75%) refer to situation 1.a in Figure 4, i.e. no scope error occurred and the relative error shown by the EM over those 12 projects was compatible with the relative error of the EM observed over the historical data. There are 3 cases out of 16 (18.75%) that refer to situation 1.b of Figure 4. This case shows that no scope error occurred, but the relative error of the EM on those 3 cases was not compatible with the RE of the EM observed over the historical data. Finally, we got only 1 case (6.25%) referring to situation 2.b of Figure 4 that we labeled by "EXT (ScE)" because it points out that a scope error occurred and the RE of the EM was not compatible with the RE of the EM observed over the historical data. We did not get any case of situation 3 in Figure 4. Overall, these results were predictable because we estimated 16 NASA projects belonging to the test set based upon 77 NASA projects which came from substantially the same context and hence the EM scope was expected to be the same.

Consequently, as discussed in Section 7, we changed the model by adding a new variable (i.e. MODE). This change brought about an overall shrinkage of magnitudes of the Error Bayesian Prediction Intervals, shrinking the EM uncertainty and increasing the number of outliers. It means that, the RE compatibility of an EM between the RE on the historical data and the new ones decreases as the number of variables increases. This is especially true when the added variables are categorical. There is a mathematical reason for this behavior. In fact, the more variables we consider, the more projects differ from each other. For instance, consider two projects P1 and P2. Assume that the software systems stemming from them have similar size, e.g., ≈ 100 KSLOC. If we considered size only, we may argue that P1 and P2 are similar to each other, but if we considered one more variable, e.g., complexity (with P1 = "Very High" and P2 = "Low"), we may argue that P1 and P2 would be very different because of the complexity.

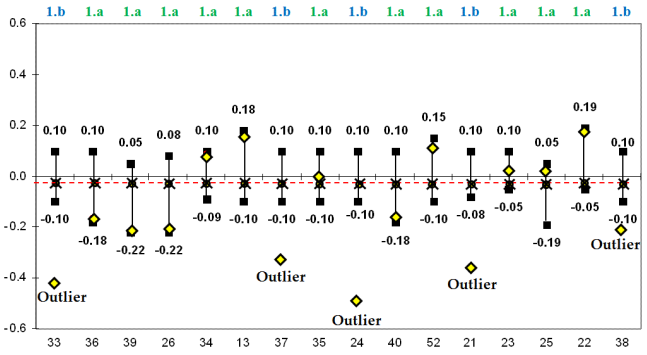

Figure 6. Error BPIs concerning *LogLinMode*.

With respect to Figure 6 (*LogLinMode* model), 11 cases out of 16 (68.75%) refer to situation 1.a in Figure 4, i.e. no scope error occurred and the relative error shown by the EM over those 11 projects was compatible with the relative error of the EM observed over the historical data used for calibrating the EM. There were also got 5 cases out of 16 (31.25%) that refer to situation 1.b of Figure 4. This case shows that no scope error occurred, but the relative error of the EM on those 5 cases was not compatible with the RE of the EM observed over the historical data. We did not get any cases like situations 2.b and 3 in Figure 4, i.e. no scope error occurred for the estimates provided by *LogLinMode*.

We can compare *LogLin* and *LogLinMode* from a scope error point of view, as well. This can be done by considering their performances. In particular, *LogLin* (75%) is able to provide a higher compatibility between the training set and test set than *LogLinMode* (68.75%). This trend is confirmed by the greater number of outliers of *LogLin* than the ones provided by *LogLinMode*. Nevertheless, *LogLin* presents a scope error on project 38, while *LogLinMode* does not provide any estimate affected by scope error. The conclusion is that, from a scope error point of view both models could be safely used for estimating all of the projects apart from *LogLin* on project no. 38 where a scope error occurred. Nevertheless, from an uncertainty point of view *LogLinMode* provided narrower intervals [21].

Based upon analysis in Figure 4, we show that, data points that we called outlier in Figure 5 and Figure 6 behave as outliers. To prove that such a statement is correct, we considered *LogLinMode* whose error BPIs are shown in Figure 6. Then, we rebuilt this model by including the outliers shown in Figure 6 (i.e., 33, 37, 24, 21, and 38). The result is reported in Figure 7.

From an accuracy point of view, the model in Figure 6 is less accurate than the one in Figure 7 because the former has $\text{Mean}(RE_{TestSet})$ = -0.026, $\text{STD}(RE_{TestSet})$ = 0.049, and $MMRE_{TestSet}$ = 0.044, where $MMRE_{TestSet} \equiv \text{Mean}(|RE_{TestSet}|)$, while the latter has $\text{Mean}(RE_{TestSet})$ = -0.007, $\text{STD}(RE_{TestSet})$ = 0.045, and $MMRE_{TestSet}$ = 0.035. This result was predictable because five projects in Figure 7 (i.e. the outliers of Figure 6)

were included in the estimation model training. Therefore, only 11 projects out of 16 were independent from the remaining projects.

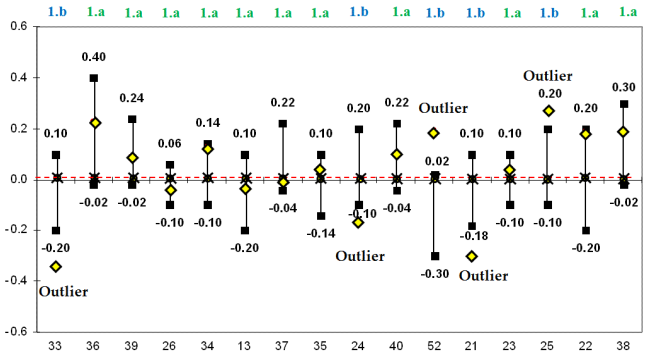

Figure 7. Error BPIs for *LogLinMode* trained with five more outliers in Figure 6.

Table 4 shows the relative errors of both models (i.e. models in Figure 6 and Figure 7) on those projects.

From an uncertainty point of view, however, the effect of including outliers in the training set of the model in Figure 7 made the uncertainty worse as we expected (compare the magnitude of the intervals of both figures). In particular, the intervals on projects 36, 52, 22, and 38 increased insofar as their magnitude exceeded the threshold of RE = 0.3 that usually is considered a suitable threshold to assess the acceptability.

Table 4. Relative error comparison.

EM in Fig. 6	-0.42	-0.32	-0.49	-0.36	-0.21
EM in Fig. 7	-0.34	0.00	-0.18	-0.31	0.19
Project	33	37	24	21	38

Note that, both models have 5 outliers, three in common (33, 24, and 21) and two different (52 and 25 in Figure 6, and 37 and 38 in Figure 7). The log-linear model with dummy variables is unable to explain these points because of the model error.

With respect to Figure 6, we also rebuilt the log-linear estimation model with categories by not including the outliers and using the remaining projects, i.e. 36, 39, 26, 34, 13, 35, 40, 52, 23, 25, and 22. The result is shown in Figure 8 that shows what we expected, i.e. the magnitude of the error prediction intervals did not increase with respect to Figure 6. Nevertheless, from an accuracy point of view, the model in Figure 8 did not improve. In fact, it has Mean($RE_{TestSet}$) = -0.024, STD($RE_{TestSet}$) = 0.198, and $MMRE_{TestSet}$ = 0.146.

Note that, the model in **Error! Not a valid bookmark self-reference.** has three identical outliers to the model in Figure 6 (33, 24, and 21). It has three other outliers (36, 52, and 25), and but does not have two outliers (37 and 38) from the model in Figure 6. The outliers in Figure 8 are the same as the ones in Figure 7, apart from project 36, where the interval did not include the actual RE. This means that, actually the log-linear model with dummy variables is not able to explain these points confirming the conclusions made above. This case study shows that, rebuilding the model with non-outliers does not increase the uncertainty and rebuilding the model with outliers does not worsen the uncertainty. Therefore, we infer that what we called outliers actually behave as what statisticians call outliers.

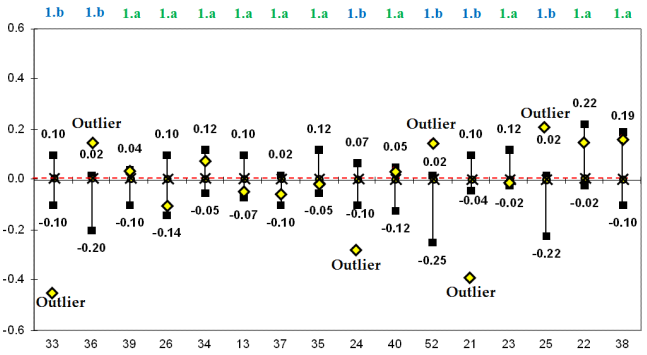

Figure 8. Error BPIs for *LogLinMode* trained with eleven more data points (non-outliers) in Figure 6.

The final result of this case study is that, if we do not know the right variables and shape of the estimation model (as is usual in real cases), the only thing we can do to use the model safely is to avoid using it on projects where a scope error can occur. To detect where a scope error can occur, we need computational tools, as the ones we presented in previous sections, in addition to the estimation model, which can be also used to extend the estimation model scope and its reliability.

10. Conclusion

In this work we presented a technique for detecting and handling scope error as well as for making a decision whether or not including outliers into the training set for rebuilding a newer version of the estimation model. The technique of detecting and handling scope error consists of analyzing the performance of estimation models from an uncertainty point of view. Based upon the method of estimating uncertainty defined by the authors, we check whether the expected relative error of the EM in estimating a project falls within the expected uncertainty interval. If it does, there is no scope error. Conversely, if the expected relative error falls outside the uncertainty interval, a scope error may happen. Once we know the actual values of the project, we can figure out whether the scope error affected the estimates.

It is important to note that, the scope error detection and analysis technique proposed in this work can be considered as a pre-condition to the safe use of the estimation model. In other words, we suggest using this technique in real cases to make sure that the estimation model is safe and reliable for estimating the considered project. Once project managers and practitioners know this information, they can make more informed decisions as to whether to use the model anyway, improving the model by changing variables and/or shape, or even whether to replace the estimation methodology. An interesting point is that, the proposed technique can be used to support any kind of estimation model, including human judgment-based approaches.

Finally, it is worth noting that, the outlier analysis presented in this work can also be used to drop those data points that make the estimation model worse in terms of accuracy and uncertainty. This means that, once we build the BDF we can use it for selecting those data points that are less risky for building a new version of the estimation model, dropping the ones that deteriorate the model. Of course, this criterion can be used to remove from the data set old data points, as well. The valuable result is that,

applying the specified analysis, we can get a less risky and more reliable model over time.

Acknowledgements: This work was supported in part by NSF Grant CCF-0438933 Flexible High Quality Design for Software. We would like to thank the University of Bolzano-Bozen and the Italian Army General Staff - Financial Planning and Budgeting Division.

11. References

[1] L. Angelis, I. Stamelios, "A Simulation Tool for efficient analogy based cost estimation," Empirical Software Engineering, Vol. 5, no. 1, pp. 35-68, March 2000.

[2] V. R. Basili, G. Caldiera, H. D. Rombach, "The Experience Factory," In Encyclopedia of Software Engineering, Ed. J.J. Marciniak, John Wiley & Sons, 1994.

[3] V. R. Basili, G. Caldiera, H. D. Rombach, "Goal Question Metric Paradigm," In Encyclopedia of Software Engineering, Ed. J.J. Marciniak, John Wiley & Sons, 1994.

[4] C. Bishop, "Neural Network for Pattern Recognition." Oxford University Press, 1995.

[5] B.W. Boehm, "Software Engineering Economics," Prentice Hall, 1981.

[6] L.C. Briand, T. Langley, and I. Wieczorek, "A Replicated Assessment and Comparison of Common Software Cost Modeling Techniques," Proc. Int'l Conf. Software Eng. (ICSE 22), pp. 377-386, 2000.

[7] The COCOMO II Suite, http://sunset.usc. edu/research/cocomosuite/index.html, 2004.

[8] S.D. Conte, H.E. Dunsmore, and V.Y. Shen, "Software Engineering Metrics and Models," Benjamin-Cummings, Menlo Park CA, 1986.

[9] B. Efron, and R.J. Tibshirani, "An Introduction to the Bootstrap." Chapman & Hall, NY, 1993.

[10] D. Husmeier, R. Dybowski, and S. Roberts, "Probabilistic Modeling in Bioinformatics and Medical Informatics". Springer, 2004.

[11] M. Jørgensen, "Experience With the Accuracy of Software Maintenance Task Effort Prediction Models," IEEE TSE, 21(8), pp. 674-681, August 1995.

[12] M. Jørgensen and D.I.K. Sjøberg, "An Effort Prediction Interval Approach Based on the Empirical Distribution of Previous Estimation Accuracy" Journal of Information Software and Technologies 45: 123-136, 2003.

[13] B. Kitchenham and S. Linkman, "Estimates, Uncertainty, and Risk." IEEE Software, 14(3), pp. 69-74, May-June 1997.

[14] B. Kitchenham, S. MacDonell, L. Pickard, and M. Shepperd, "What accuracy statistics really measure," IEE Proceedings - Software Engineering, 48, pp81-85, 2001.

[15] D.J.C. MacKey, "Bayesian Models for Adaptive Models." Ph.D. Thesis, California Institute of Technology, Pasadena, CA, USA, 1991.

[16] S. McConnell, "Software Estimation, Demystifying the Black Art" Microsoft press, 2006.

[17] T. Menzies, D. Port, Z. Chen, J. Hihn, "Validation Methods for Calibrating Software Effort Models", Proc. 27st Int. Conf. Software Eng. (ICSE), 2005.

[18] I. Myrtveit, E. Stensrud, and M. Shepperd, "Reliability and Validity in Comparative Studies of Software Prediction Models," IEEE Trans. Software Eng., vol. 31, no. 5, pp. 380-391, May 2005.

[19] D. Port and M. Korte, "Comparative Studies of the Model Evaluation Criterions MMRE and PRED on Software Cost Estimation Research", ACM, (ESEM08) Liepzig, Germany, 2008, pp. 63- 70.

[20] S.A. Sarcia', V.R. Basili and G. Cantone , "An Approach to Improving Parametric Estimation Models in the Case of Violation of Assumptions Based on Risk Analysis", CS-TR-4928, UMIACS-TR-2008-20 University of Maryland (USA), 2008.

[21] S.A. Sarcia', G. Cantone, and V.R. Basili, "Using Uncertainty as a Model Selection and Comparison Criterion" ACM, PROMISE 2009, Vancouver, CA, 2009.

[22] M. Shepperd, "Software project economics: a roadmap," FOSE'07, IEEE, 2007.

[23] J.S. Shirabad, and T. Menzies, "The PROMISE Repository of Software Engineering Databases," School of Information Technology and Engineering, University of Ottawa, Canada. http://promise.site.uottawa.ca /SERepository, 2005.

[24] S. Weisberg, "Applied Linear Regression", 2nd Ed., John Wiley and Sons, NY, 1985.