Using Uncertainty as a Model Selection and Comparison Criterion

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
Dept. of Computer Science, University of 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, software estimation research has been searching for the best model for estimating variables of interest (e.g., cost, defects, and fault proneness). This research effort has not lead to a common agreement. One problem is that, they have been using accuracy as the basis for selection and comparison. But accuracy is not invariant; it depends on the test sample, the error measure, and the chosen error statistics (e.g., MMRE, PRED, Mean and Standard Deviation of error samples). Ideally, we would like an invariant criterion. In this paper, we show that uncertainty can be used as an invariant criterion to figure out which estimation model should be preferred over others. The majority of this work is empirically based, applying Bayesian prediction intervals to some COCOMO model variations with respect to a publicly available cost estimation data set coming from the PROMISE repository.

Categories and Subject Descriptors
D.2.9 [**Cost Estimation**]: Empirical study as to software cost estimation model evaluation criterion.

General Terms
Management, Measurement, Performance, Reliability, Model Evaluation, Model Selection, Accuracy, Uncertainty.

Keywords
Cost Estimation, Cost Model, Prediction Interval, Bayesian Prediction Intervals, Model Selection, Model Evaluation, Calibration.

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). This huge research effort did not lead to a common agreement of what is best. One of the major issues is that, they have been using accuracy as the selection and comparison criterion, which is not invariant, but depends on the test sample, the error measure, and the chosen error statistics (e.g., MMRE, PRED, Mean and Standard Deviation of error samples [7]).

In this work we define a different criterion for comparing and selecting estimation models. Unlike traditional comparison approaches, the one that we propose in this paper is not based on evaluating accuracy. Instead, it is based on calculating and evaluating uncertainty.

The authors have defined a technology [20] based on analyzing prediction errors from each evolving estimation model over its history. The evolving models were obtained using an artificial neural network as each new project is estimated and possibly added to the model. In particular, it allows calculating Bayesian Prediction Intervals which quantify uncertainty more efficiently and effectively than previous methodologies. The proposed approach allows avoiding any specific assumption on the model and provides much smaller prediction intervals than the ones obtained so far.

This paper is organized as follows: It begins with a brief discussion of related work; followed by a description of parametric estimation models and estimation model accuracy. We then define the new comparison criterion which is invariant with respect to the considered error measures and accuracy indicators. The case study using the NASA 93 COCOMO data set [19] is described and the results of applying the technology is presented, along with the research methodology and practical hints on how to use the proposed technology in real cases.

2. RELATED WORK
Literature on software estimation is amazingly large. However, it mainly refers to cost estimation. As noted by Port and Korte [18], "there exist a relatively large number of empirically based effort estimation models". However, the human-based estimation literature, apart from the works from Simula Research Labs [11], [13], [15], is relatively small.

Some authors have summarized the state of the art of this field over the last 25+ years [21], [17], [19]. All of them argued that, this huge research effort led to misleading results. Currently, there is not agreement about which estimation model is best (e.g., regression models, machine learning, predefined models such as COCOMO, COCOMO-II, and CART). The main point is that, researchers and practitioners did not agree on shared criteria of evaluation. Each organization used its own statistics (indicators), methodologies, and procedures. The consequence is that research

in this field is moving in different directions. The result is that software engineering practitioners have neither the best estimation model nor a shared criterion for evaluating an estimation model.

3. PARAMETRIC 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 $cov(\varepsilon) = \sigma^2 I$ [22], where $E(\varepsilon)$ is the expected value of ε. 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 of 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) 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.

4. ESTIMATION MODEL ACCURACY

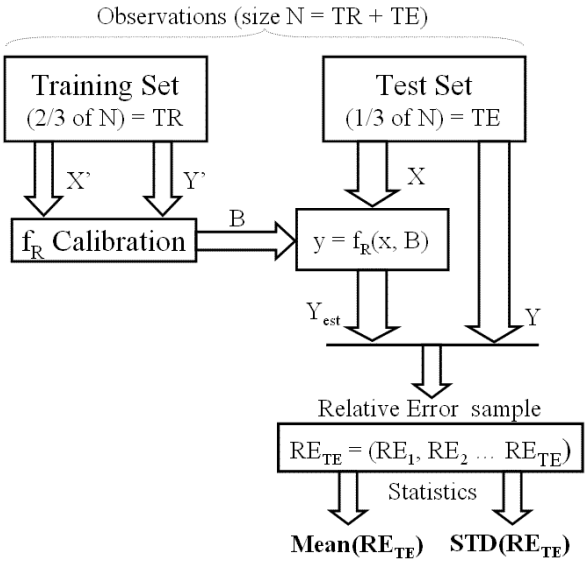
Residuals, or some related measures (e.g. relative residuals), are used to calculate the accuracy statistics (also called accuracy indicators). It means that, we may use an Absolute Error (AE), i.e. AE = Actual – Estimated = $e = Y - Y_{est}$. However, an absolute error measure makes no sense when dealing 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 cannot be used for model comparison. When predicting software engineering variables, the right choice is to consider a relative error measure, i.e. a measure that takes into account the size of what we are predicting [17]. For this reason, Boehm [4] defined the performance of his software cost model (COnstructive COst MOdel, COCOMO) in terms of *Relative Error* (RE), Eqn. 1.

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

Note that, currently the Boehm's model has been enhanced into the COCOMO-II [6], but the accuracy evaluation principles have not changed. The evaluation procedure of the model accuracy is shown in Figure 1.

We start by considering a *data set* of size N, which is split up into two subsets, a training set and a test set. The training set size is TR = (2/3)N and the test set size is TE = (1/3)N, thus N = TR + TE. We choose the splitting proportion 2/3-1/3 because it is a usual choice when dealing with hold-out methodologies [16].

Figure 1. Evaluation scheme of estimation models.



Another valid proportion is 80%-20%. Based on the training set, we calibrate function f_R. Note that, to avoid any confusion between the training set and the test set, we use an apostrophe to point out that the data belongs to the training set (i.e., X' and Y'). Once we have calculated the parameters B, we feed X into f_R and obtain the estimated values Y_{est}. Notice that, the size of Y_{est} is TE, i.e. $Y_{est} = (Y_{est}^{(1)} \dots Y_{est}^{(TE)})$. Then, accuracy is evaluated by calculating two summary statistics over the Relative Error sample as shown in Figure 1. In particular, $Mean(RE_{TE}) = (1/TE)\Sigma_{i=1..TE}(RE_i)$ is a measure of the *bias* of f_R and $STD(RE_{TE}) = SQRT\{(1/N)\Sigma_{i=1..TE}([RE_i - Mean(RE_{TE})]^2)\}$ is a measure of the *spread* of f_R. Note that, a correct evaluation can be done through both statistics at the same time. For instance, if we find out that a model has a $Mean(RE_{TE})$ closer to zero with respect to another model, we can infer that the former is more accurate than the latter in terms of *correctness*. Moreover, if the former has a narrower $STD(RE_{TE})$ than the latter, we can infer that the former is more accurate in terms of *reliability* (stability).

Therefore, two or more parametric estimation models can be compared by considering Mean and STD calculated over the same test set. Note that, each estimation model has to be calibrated based upon the same training set. The model that provides a shorter bias and spread is assumed better than the others. Note that, concept underlying Figure 1 can be used for evaluating accuracy of human-based estimation, as well [11].

Using the scheme in Figure 1 implies we are evaluating the models being selected by the *model error*, i.e., we evaluate both the suitability of the model shape (e.g., linear, geometric) and the relevance of the independent variables. However, this is not sufficient to come up with a correct evaluation. We should check whether regression assumptions hold for each model being compared (Section 3). For instance, heteroscedasticity may affect the error distribution as well, as the latter may not be Gaussian. If regression assumptions do not hold, we should consider more suitable indicators to evaluate the model, e.g., statistics that are less sensitive to outliers, non-parametric statistics, using neither STD nor Mean. The accuracy evaluation should not depend on the

indicators we use, i.e., the model selection criterion should be invariant to it. Moreover, if we selected one of the competing models through the scheme in Figure 1, we would not take into account all of the information about *assumption error* (the one that we may get from assuming wrong inputs) and *scope error* (the error about using the model in a different context from the one used to calibrated it).

Another important issue regarding the scheme in Figure 1 is that the test set on which the evaluation is made may have no relationships at all to the project being estimated. That is, applying the scheme in Figure 1 would mean that we would make a decision about which model is best without taking into account what we are trying to predict. We argue that this (traditional) approach is simply unsuitable for the goals of mature organizations [1]. We must evaluate the estimation model just over what we are predicting, even though the information on it may be affected by uncertainty. In other words, we should not select a test set independently from the project being estimated, but we should evaluate performance over it and take into account uncertainty. The reason is that the estimation model may behave differently from project to project. Therefore, if we calculated Mean and STD over an unsuitable test set, the actual estimation model performance would be masked.

There is another major point that makes scheme in Figure 1 unsuitable for selecting estimation models, the fact that organizations find it much more useful to have two-point predictions instead of one-point predictions as used in Figure 1. In fact, when dealing with real project predictions, nobody would think that one-point predictions would be the right ones. Everybody thinks that a range where the prediction may fall with a stated confidence is much more useful and credible [15]. Mathematically, such an interval is called Prediction Interval defined as the 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 Months) for the next estimate will lie between 120 and 180 with a confidence of 95%. Of course, the method for estimating such a range has to take into account any regression violations.

5. BAYESIAN PREDICTION INTERVALS

The approach we propose is an alternative to the non-parametric bootstrap method [8] and the Bayesian approach [14]. It is an extension of the Jørgensen and Sjøberg's technique for calculating empirical PIs for both regression-based models [12] and human-based judgment techniques [11]. Although the proposed methodology may be eventually be bootstrapped or included in a Markov Chain Monte Carlo simulation framework, in this initial stage, we do not consider resampling procedures or simulation approaches so as not to complicate the methodology definition. Moreover, one of the major aims in this technique was to try to find an improved approach in terms of computational cost and performance.

The proposed strategy of estimating PIs is based on removing as many regression assumptions as possible and considering a sample of relative errors, i.e., a training set composed of relative errors, RE_{TR} (Figure 1). Since we do not assume that the distribution is a Gaussian and the sample is homoscedastic, we consider the distribution asymmetric, affected by outliers, and with variable spread. Moreover, we assume RE to be x-correlated.

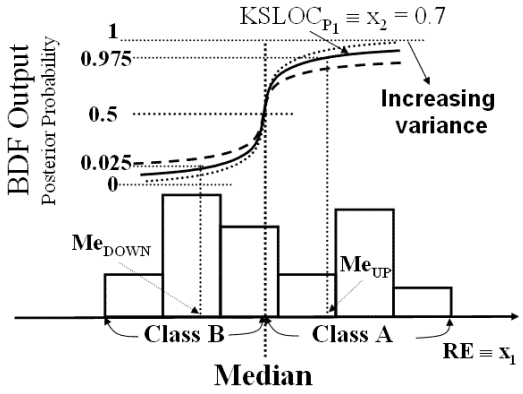
We estimate Bayesian Prediction Intervals by applying the six steps shown below.

(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 [20]. 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, …, x_N. This kind of regression function provides an x-dependent median, minimizing the Minkowski R-distance (R = 1). It is called robust regression because it is less sensitive to outliers and asymmetric distributions.

(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 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 (MLFFNND) so that its output provides a classification decision, i.e. a new data point is classified as belonging to A or B according to its similarity to the data used for training. Therefore, our "a priori" information tells us how far a point is above or below the x-dependent median (binomial choice).

Figure 2. Posterior probability density obtained by fixing KSLOC and letting RE vary. Dotted and dashed lines represent posterior probability functions with an increasing variance.



We call this MLFFNND a Bayesian Discrimination Function (BDF) because its output can be interpreted as the posterior probability that any input belongs to class A [3]. Any input is classified as belonging to class A, if the BDF output is between [0.5, 1], e.g. 0.85. It is classified as belonging to class B otherwise, i.e. the BDF output is in [0,0.5[, e.g. 0.25, where [.,.] is a closed interval and [.,.[is a right-open interval.

Actually, we are not interested in classifying our new observations. Our aim is to use the BDF for inference (the inverse problem). Making inference by the BDF in Figure 2 means selecting the interval (Me_{DOWN}, Me_{UP}) on the x-axis corresponding to the two fixed confidence limits (e.g. [0.025, 0.975]). 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.

The defined BDF in Figure 2 is expressed by the following relationship, $f_{BDF}(x_1 = RE, x_2=KSLOC) = [0,1]$, i.e. $y = f_{BDF}(x_1, x_2) = Pr(y=1|x_1, x_2)$, where [0, 1] points out any real number in [0,1] (e.g. the posterior probability), x_1 and x_2 represent the sample information, and y=1 represents class A (y=0 represents class B). Assume that, the BDF yields $f_{BDF}(P_1) = 0.85$ and $f_{BDF}(P_2) = 0.25$. Then, project P_1 is classified as belonging to class A, because $f_{BDF}(P_1) \geq 0.5$ and project P_2 is classified as belonging to class B because $f_{BDF}(P_2) < 0.5$.

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. Then, BDF turns into $Pr_{x_1}(y=1|x_2) = f_{|x_2=c}(x_1) = y(x_1)$. Note that, in the case of an N-dimensional space (with N > 1), we fix all variables except x_1 (the relative error RE), i.e. $Pr_{x_1}(y=1|x_2.x_N) = f_{|x_2=c_1,...x_N=c_{N-1}}(x_1) = y(x_1)$ where the variables (x_2 … x_N) are the same as the independent variables of the estimation model.

(4) Once we build the posterior probability density (solid line in Figure 2), we can obtain a Bayesian PI by fixing a 95% confidence level, i.e. (0.025, 0.975), and picking 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. The posterior probability density in Figure 2 has an important characteristic. Its slope gets steeper as the variance decreases; it gets flatter as the variance increases [9]. Therefore, the increasing variance of the relative error with respect to the independent variables of the model has a geometric representation. It corresponds to the flatter slope of the sigmoid curves in Figure 2, e.g. the variance corresponding to the dotted line is lower than the variance corresponding to the dashed line. Therefore, the proposed strategy based on the BDF is able to evaluate the variance (calculating the RE range) from the slope of the posterior probability density function empirically. This approach is quite different from estimating the variance by resampling procedures such as bootstrap. See [9, pp. 20-24] for additional explanations.

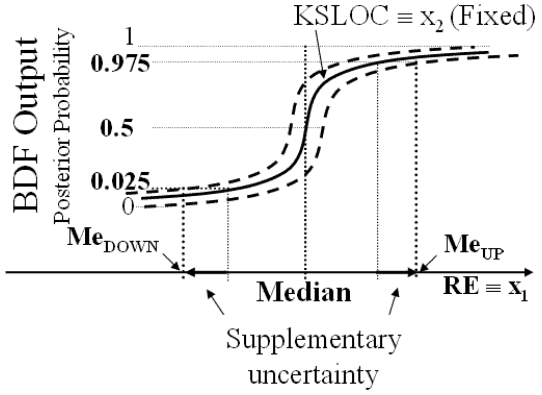
(5) To calculate the estimation PI (e.g. the effort) for the RE range, i.e. (Me_{DOWN}, Me_{UP}), we first consider the formula RE = (Actual − Estimated)/Actual and then, we deduce Actual = Estimated/(1 − RE).

As shown by Jørgensen et al. [11], the PI is $[O_{est}^{N+1}/(1-Me_{DOWN}), O_{est}^{N+1}/(1-Me_{UP})]$, where $O_{est}^{N+1} = f_R(x', b)$, e.g. $f_R(x' = 0.7, b)$. For instance, assuming that the RE interval obtained from Figure 2 is [-0.9, 0.1] and Estimated effort = 3 person months, the PI is [3/(1-(-0.9)), 3/(1-0.1)] =[1.6, 3.4] person months.

Although the proposed prediction interval shown in Figure 2, 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, MacKey proposes to apply the Bayesian framework to classification problems. 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 3. 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.



Since the leave-one-out cross-validation (LOOCV) score is calculated by the SQRT(MSE), which is an unbiased estimator of the generalization error of the BDF, we use this quantity as a correction factor. Instead of using LOOCV, we may apply K-fold CV, as well. However, its score would not be an unbiased estimator of the generalization error even though it would be more realistic than the score obtained by leaving out only one data point. We sum and subtract the LOOCV score to the posterior probability density function (solid line in Figure 2), obtaining the two outer dashed lines in Figure 3. In particular, the upper dashed line is $f_{|x_2=c}(x_1) + SQRT(MSE)$ and the lower dashed line is $f_{|x_2=c}(x_1) - SQRT(MSE)$. The band included within the two dashed lines represents the overall uncertainty due to the variation in the BDF parameters. 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.

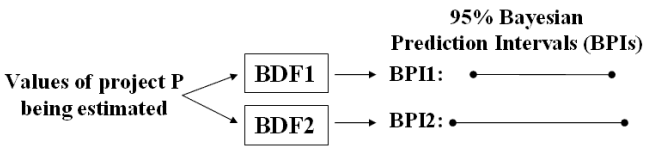
6. INVARIANT COMPARISON CRITERION

In this research, we propose a model selection criterion, which uses uncertainty instead of accuracy. The aim is to make more informed decisions about which model is best with respect to the project being estimated. We argue that the best model is the least risky one. Therefore, unlike traditional approaches, we take into account any regression model violation, assumption error, and scope error. As explained in Section 4, the approaches applied so far quantify uncertainty using traditional Prediction Intervals (i.e., based upon the frequentist paradigm). However, the authors have recently showed that traditional approaches for calculating prediction intervals cannot be effectively used to achieve this goal [20] because the magnitude that they provide is too wide to be of any utility. Moreover, traditional approaches do not take into account either regression violations or assumption and scope errors, while real world phenomena hardly ever satisfy regression assumptions and estimates often display scope and assumption errors.

The strategy consists of using Bayesian Discrimination Functions (BDF) as explained in Section 6 for each considered estimation model/method being compared. The BDF is actually an "intelligent" function that is built upon the (relative) errors coming from the use of the estimation model over a sample set of observations. Based on the approach proposed by Jørgensen and Sjøberg [11], once we get an Error Bayesian Prediction Interval, we can turn it into an Estimate Bayesian Prediction Interval, as explained in Section 5. Note that, Error Bayesian Prediction Intervals and Estimate Bayesian Prediction Intervals are completely interchangeable hence they can both be used for comparison. We explain why in the following paragraph.

As specified above, for comparison purposes, we can use either Error Prediction Intervals (i.e. [RE_{DOWN}, RE_{UP}]) or Estimate Prediction Intervals (i.e. [Act_{DOWN}, Act_{UP}]) because the magnitude of Error Prediction Intervals keeps proportionally constant with respect to the magnitude of Estimate Prediction Intervals, and vice versa. Usually, when dealing with numbers, we prefer using Error Prediction Intervals because of their smaller magnitudes. Nevertheless, because relative errors can be negative values, when not dealing with the numbers but concepts, we prefer to use Estimate Prediction Intervals because they seem to be easier and quicker to grasp. Therefore, our conceptual explanation will be given by considering Estimate Prediction Intervals (all of the quantities are positive), while the case study will be set up based upon a (Relative) Error Prediction Interval (quantities can be negative as well).

Figure 4. Bayesian Prediction Interval-based comparison of estimation models.



Based on Figure 4, if we compare only two estimation models, EM1 and EM2 (e.g., EM1 = Machine learning and EM2 = COCOMO II), we have to build two different BDFs, i.e. BDF1 and BDF2.

Applying the technology defined in Sarcia et al. [20] has the benefit of using the Bayesian approach, which allows shrinking the magnitude of the prediction intervals, making them more attractive and useful than the traditional prediction intervals.

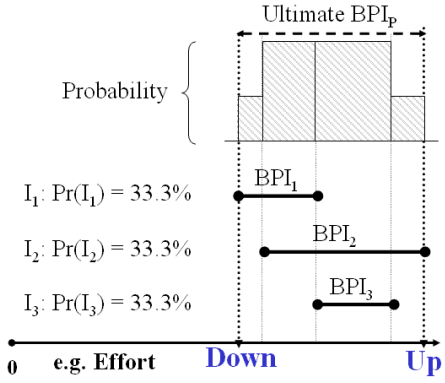
Through BDFs, we get Bayesian Prediction Intervals, which have the capability of dealing with the consequences of regression violations and displaying the scope error. A scope error is highlighted when the BDF, fed with values of the project being estimated, does not provide any BPI.

Once we know that an estimation model provides a scope error on project P (i.e. observations used for calibrating the model are so different from project P that the model may have unexpected behaviors), we should decide whether or not to use such a model for prediction. Nevertheless, we suggest discarding all of the models showing a scope error and keeping the ones that provide acceptable intervals. If none of the models provide valid intervals, we suggest considering further models.

The comparison brought out in Figure 4 is not complete. Before making a decision whether model EM1 is better/worse than EM2, we have to take into account the assumption error, as well. Assumption error is about whether we are sure about the actual values of project P being estimated. In other words, there are different possible values representing project P that we should consider. For instance, we may have C possible inputs for project P, i.e., $I = (I_1, I_2, ..., I_C)$ each having its own probability to occur $Pr(I_1), Pr(I_2), ..., Pr(I_C)$. If we do not know those probabilities, the uncertainty is on its maximum, i.e., $Pr(I_1) = Pr(I_2) = ... = Pr(I_C) = 1/C$. For instance, uncertainty may be about the project size or the project complexity. If there is not uncertainty about the project characteristics, there is no assumption error and I is composed of only one set. In that case, comparison in Figure 4 would be complete.

To explain the way of dealing with the assumption error, consider the example in Figure 5.

Figure 5. Calculating the Ultimate Bayesian Prediction Interval (UBPI) for project P being estimated. Project P has three possible input sets (I_1, I_2, and I_3) each having probability of 0.33. Note that, magnitude of the UBPI cannot be shorter than the magnitude of each individual BPI.

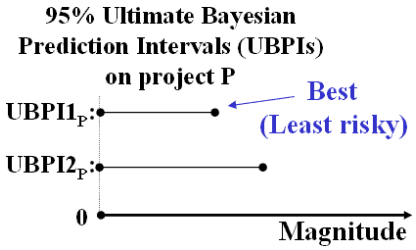


The overall uncertainty in the case of an assumption error is the union of all of the individual Bayesian Prediction Intervals. We call this union as Ultimate Bayesian Prediction Interval (UBPI). The output of applying the strategy in Figure 5 is then $UBPI_P$, i.e. the UBPI on project P. It has to be calculated for each Estimation

Model/Bayesian Discrimination Function. For instance, if we have only two estimation models to compare as in Figure 4, we get two different Ultimate BPIs on project P, say $UBPI1_P$ and $UBPI2_P$ corresponding to model 1 (EM1) and model 2 (EM2), respectively.

Going back to Figure 4, we can now perform the comparison between models EM1 and EM2 (Figure 6).

Figure 6. Ultimate Bayesian Prediction Interval-based comparison of two estimation models EM1 and EM2. The comparison is made through values of project P



The result of the comparison is that, the best model is the one that shows the shortest Ultimate Bayesian Prediction Interval magnitude, i.e., $UBPI1_P$ in Figure 6. Therefore, we consider the best model to be the least risky model among those considered (i.e., showing the least uncertainty in estimating project P). Note that, if we had more than two models, the strategy in Figure 6 would not change.

If we used the Error Bayesian Prediction Interval instead of the Estimate Bayesian Prediction Intervals, where negative values of errors may occur, we should calculate the absolute value to obtain the magnitude.

7. THE CASE STUDY

In this section, we apply the strategy presented in Section 6 to the NASA COCOMO data set [19] by comparing different models having different features and shapes from each other. For the reasons stated in Section 6, the comparison is performed by an Error Bayesian Prediction Interval instead of an Estimate Bayesian Prediction Interval. The analysis presented in this section aims at demonstrating the application of the proposed approach using real data.

Consider the situation where NASA is the learning organization [1] using measurement [2] 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. NASA's goal is to figure out which model is best for estimating the effort of the 16 next software systems (from 1985 to 1987). Since we actually know the data from these 16 software systems, we can use them as a test set for evaluating the proposed comparison strategy over 16 cases.

The data set [19] is composed of 93 project instances. Each instance is described by 24 attributes (Table 1). In particular, "Size", 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 [4].

Based upon data from the 77 software systems, our case study consists of calibrating 4 different models and comparing them to each other through both the traditional criterion and the one we propose in this research (uncertainty). Conclusions enacted from this case study offer some new insights into the projects developed at NASA and show that using uncertainty as an invariant model comparison criterion can be a useful approach for making more informed decisions and increasing the prediction capabilities of mature organizations.

Table 1: 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:

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

Note that, the right way of coding categorical values is to use dummy (dichotomous) variables. If we have C categorical values, we create (C – 1) dichotomous variables, i.e. the variable can have values 0 or 1. Since "Mode" has three categories ("Semidetached", "Embedded", "Organic"), we included two dummy variables (D1, D2) in both *LinMode* and *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.

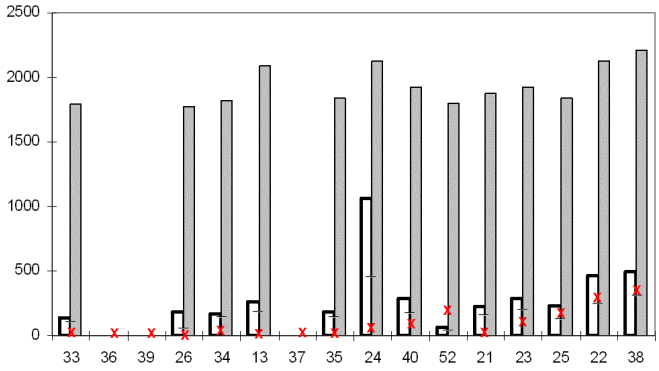
We only used the attribute "YEAR" to split up projects, i.e. the first set (before 1985) was composed of 77 software systems, and the second set (after 1984) was composed of 16 software systems.

To focus the goal of our study, we defined a **GQM template**: *Analyze* the uncertainty of regression models *for the purpose of* comparison *with respect to* the magnitude of Error Bayesian Prediction Intervals from the *point of view* of the project managers in the *context of* NASA's projects.

8. RESULTS

Before presenting the results, we show that the BDF-based technique [20] provides more useful Prediction Intervals than the traditional approach. In particular, Error Bayesian Prediction Intervals (ErBPIs) are much shorter than the traditional ones. Of course, it would be the same if we used Estimate Bayesian Prediction Intervals (EsBPIs). To this aim, we only considered the function *Lin* and calibrated the BDF by applying the BDF-based procedure [20]. Then, we fed each element in the test set (16 data points) into the BDF and obtained 16 ErBPIs. Consequently, we turned the ErBPIs into the EsBPIs (unfilled rectangles in Figure 7) applying the procedure explained in Section 6. We also calculated Prediction Intervals by the tradition approach (filled rectangle in Figure 7). Since the traditional approach provided negative values for the lower limit of the rectangles in Figure 7, we decided to put the lower limit of these rectangles to zero.

Figure 7. Comparison of magnitudes between Bayesian Prediction Intervals (filled) and the Traditional ones (unfilled).



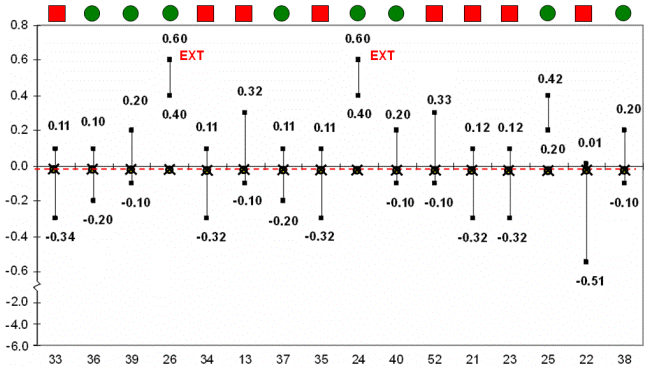
The y axis of the diagram in Figure 7 represents the effort and the x-axis expresses the software system belonging to the test set. The number shown below the x-axis in Figure 7 is the ID assigned in the original data set from NASA [19]. An "X" stands for actual effort and a "–" stands for estimated effort. Apart from the projects on which function *Lin* provides negative effort values (i.e., ID = 36, 39, and 37) where the analysis makes no sense, the Bayesian approach provides shorter prediction intervals in terms of magnitude than the traditional one for all of the cases. With respect to project 52, however, the interval does not include the actual RE meaning that occasionally the interval may be biased. Based on data in Figure 8, a valuable result is that the proposed approach is a valid alternative to the traditional methodology for shrinking the estimate prediction intervals. It makes possible using uncertainty for comparison purposes.

To carry out a comparison among the four considered estimation models, we first calculated the BDF for each model. Then, we fed the test set (remaining 16 software systems) into each BDF and obtained 16 Error Bayesian Prediction Intervals for each estimation model (Figures 8, 9, 10, and 11) . Unlike Figure 7, we did not turn ErBPIs into EsBPIs, but used the magnitude of the ErBPIs for comparing the models.

With respect to Figure 8, 9, 10, and 11, the y-axis is the relative error (RE) and in the x-axis there are 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. "EXT" refers to the fact that, the ErBPI does not include the median of the relative error. Therefore, as we explained in Section 6, it may signal a possible scope error. Moreover, we used a circle at the top of each diagram to indicate that the magnitude of the related Bayesian Prediction Interval exceeded the value of 0.30, assumed as an acceptable error magnitude. We used a square to indicate an error magnitude exceeding the value of 0.30, assumed as an unacceptable error magnitude. The limit of 0.30 plays the same role of H in the indicator PRED(H) = X. In other words, we may also consider the percentage of the number of circles (success) with respect to the overall number of elements in the test set as an overall uncertainty measure.

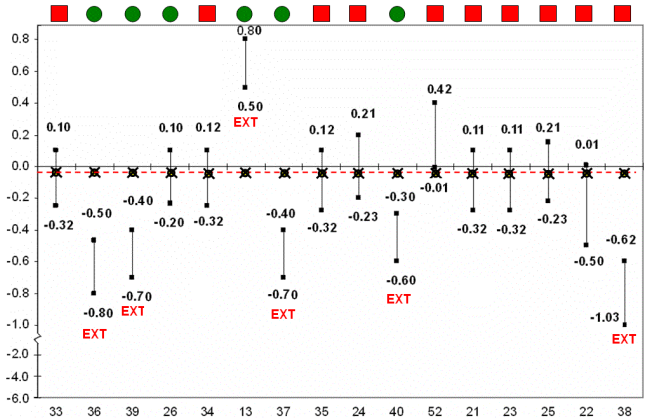
For instance, in Figure 8 (model *Lin*), the Bayesian Prediction Interval PRED (BPI_PRED) would be BPI_PRED(0.30) = 8/16 = 50%. Of course, 50% is not a good result, showing that the linear model does not perform very well. With respect to Figure 8, we have projects 26 and 24 on which a scope error may occur. Note that, as with PRED(0.25) and PRED(0.20), we can also consider a BPI_PRED(0.25) or even a BPI_PRED(0.20), respectively.

Figure 8. Error Bayesian Prediction intervals for the model *Lin*.



With respect to Figure 9 (model *LinMode*), we can see that adding the categorical variable *Mode* did not shrink the intervals, but they even increased in magnitude. BPI_PRED(0.30) = 6/16 = 37.5%, therefore we concluded that *LinMode* performed worse than *Lin* in terms of uncertainty. *LinMode* showed 6 projects in which a scope error may occur, as well.

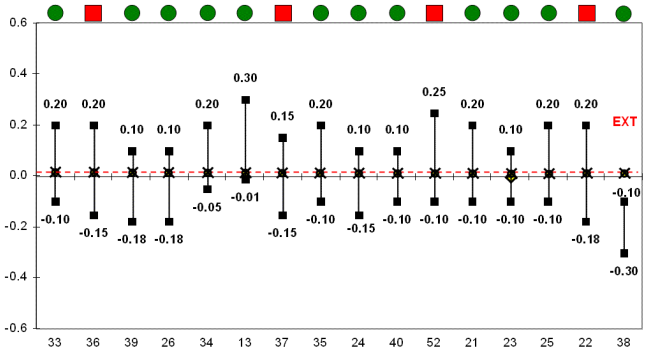
Figure 9. Error Bayesian Prediction intervals for the model *LinMode*.



With respect to Figure 10 (*LogLin*), we can see that the logarithmic transformation definitely improved the model. In fact, BPI_PRED(0.30) = 12/16 = 75% with only one "EXT" on project

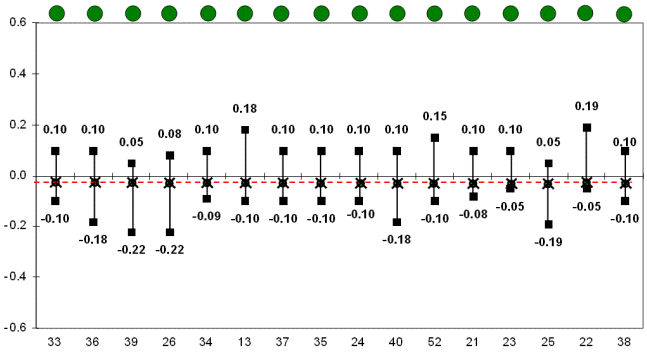
38. Result in Figure 10 was expected somehow, because we know that a geometric function correctly describes the relationships between Effort and COCOMO variables.

Figure 10. Error Bayesian Prediction intervals for the model *LogLin*.



With respect to Figure 11 (*LogLinMode*), we included the categorical variable *Mode* in the logarithmic model. Unlike Figure 9, where the variable *Mode* made the model worse, in this case we got an improvement in terms of uncertainty, i.e., the ErBPIs shrank. In fact, BPI_PRED(0.30) = 16/16 = 100% without "EXT" points.

Figure 11. Error Bayesian Prediction intervals for the model *LogLinMode*.



From the accuracy point of view, however, the situation is different. As shown in Table 2, the *LogLin*, model is slightly more accurate than the *LogLinMode* model..

Table 2: Comparing the models with respect to accuracy.

	Lin	*LinMode*	*LogLin*	*LogLinMode*
Mean(RE)	-0.525	-0.740	-0.005	-0.026
STD(RE)	3.115	2.751	0.045	0.049
MMRE	2.396	2.214	0.030	0.044
PRED(0.30)	6.25%	12.50%	100.00%	100.00%

This is a very interesting situation because, based on the proposed analysis, we show that before selecting a model as best we should take into account uncertainty, as well. In particular, our analysis says that NASA should use *LogLinMode* if they care about uncertainty, while NASA should use *LogLin* if they care about accuracy.

The comparison criterion that we presented in Section 6 refers to comparing the models over each project being estimated (16 data points). Therefore, we have to make a decision as to whether or not to select a specific estimation model for prediction based upon its uncertainty with respect to the project being estimated. To this end, Table 3 shows the results of comparing, project by project, the four models under study.

Table 3: Comparing, project by project, the models with respect to uncertainty.

ID:	33	30	39	26	34	13	37	35
Lin	0.45	0.30	0.30	0.20	0.43	0.42	0.31	0.43
LinMode	0.42	0.30	0.30	0.30	0.44	0.30	0.30	0.44
LogLin	0.30	0.35	0.28	**0.28**	0.25	0.31	0.30	0.30
LogLinMode	**0.20**	**0.28**	**0.27**	0.30	**0.19**	**0.28**	**0.20**	**0.20**

ID:	24	40	52	21	23	25	22	30
Lin	0.20	0.30	0.43	0.44	0.20	0.22	0.52	0.30
LinMode	0.44	0.30	0.43	0.43	0.43	0.44	0.51	0.41
LogLin	0.25	**0.20**	0.35	0.30	0.20	0.30	0.38	**0.20**
LogLinMode	**0.20**	0.28	**0.25**	**0.18**	**0.15**	**0.15**	**0.24**	**0.20**

In particular, the filled rectangles including bold figures show the smallest magnitude among the models. Therefore, we concluded that, from a practical point of view, based upon the comparison criterion defined in this work, the best model was *LogLinMode* on projects 33, 30, 39, 34, 13, 37, 35, 24, 52, 21, 23, 25, 22. With respect to projects 26 and 40 the best (least risky) model was *LogLin*. With respect to project 30 the best model was either *LogLin* or *LogLinMode*.

9. CONCLUSION

The uncertainty analysis presented in this work has an important implication for software organizations and practitioners in terms of model comparisons. Analyses presented in this work show that, one estimation model can be more accurate than another, even though the former may be more risky than the latter. Unlike traditional approaches, the criterion presented in this paper is invariant with respect to the error measures and indicators for making the comparison.

Before claiming that one model is "better" than another, however, an organization should specify the nature of the comparison and the context in which the comparison is made. For instance, one should say that one model is better than another in terms of accuracy (or in terms of uncertainty/risk) and select the one that is more appropriate with respect to organization's goals. Nevertheless, accuracy criteria are not invariant. In the example shown above, an organization aiming at accuracy should select the log-linear model without categories (*LogLin*). An organization aiming at shrinking the uncertainty should select the log-linear model with categories (*LogLinMode*). It is worth noting that, some authors [15], [11], however, argue that providing one-point estimates (i.e. aiming at accuracy) is ineffective and even misleading. The authors believe that organizations should aim at uncertainty, i.e., estimates should always be provided in terms of prediction intervals (two-point estimates) because two-point estimates are more realistic and useful for software organizations.

Based on the uncertainty analysis, before predicting a new project, we should consider different models as stated above and select the one that provides the least risk. Once the most accurate and least risky model has been selected among those considered, we can use the model to predict predicting new projects and continue to evolve the model to find the best model repeatedly.

10. REFERENCES

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

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

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

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

[5] Briand, L.C., El-Emam, K., Maxwell, K., Surmann, D., and Wieczorek, I. 1999. An Assessment and Comparison of Common Cost Software Project Estimation Methods. Proc. 21st Int'l Conf. Software Eng. (ICSE 21), pp. 313-322.

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

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

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

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

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

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

[12] Jørgensen, M. 2004. Regression Models of Software Development Effort Estimation Accuracy and Bias. Empirical Software Engineering, Vol. 9, 297-314.

[13] Jørgensen, M., Teigen, K., and Moløkken, K. 2004. Better sure than safe? Overconfidence in judgment based software development effort prediction intervals. J. of Systems & Software, 70, pp79-93.

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

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

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

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

[18] Port D. and Korte, M. 2008. Comparative Studies of the Model Evaluation Criterions MMRE and PRED on Software Cost Estimation Research. ACM, (PROMISE'08) Liepzig, Germany, pp. 63- 70.

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

[20] Sarcia', S.A., Basili, V.R. and Cantone, G. 2008. 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).

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

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