Discussion Paper
5/2002

SW Cost Estimation: Measuring Model Performance of Arbitrary Function Approximators

Erik Stensrud, Ingunn Myrtveit

ABSTRACT

Estimating software development cost with high accuracy is still a largely unsolved problem. Consequently, there is ongoing, high activity in this research field; a large number of different estimation models ranging from mathematical functions to arbitrary function approximators (AFA's) have been proposed over the last 20+ years. Unfortunately, the studies do not converge with respect to the question "which model is best?" when functions and AFA's are compared. So far, it has not been understood why this is so. In this empirical study, we show that this is due to inappropriate validation methods as far as the validation of AFA's is concerned. In fact, the de facto validation method, cross-validation combined with MMRE, will give completely arbitrary results for AFA's. Obviously, other criteria are called for in order to appropriately assess the performance of AFA's. This should be a topic of future research.

1. INTRODUCTION

Estimating software development cost with high accuracy is still a largely unsolved problem. Consequently, there is ongoing, high activity in this research field. A large number of different estimation models ranging from mathematical functions (e.g. regression analysis and COCOMO (www)) to arbitrary function approximators, AFA's (e.g. estimation by analogy - EBA, classification and regression trees - CART, and artificial neural networks -ANN) have therefore been proposed over the last 20+ years.

Unfortunately, the studies do not converge with respect to the question "which model is best?" Especially, there have been reported very contradictory results in studies comparing an AFA with a function. Also, the performance of AFA's varies wildly across studies.

Some studies conclude that EBA models outperform regression models (for example, Shepperd and Schofield, 1997). Other studies find the exactly opposite result, namely that regression models are superior to EBA models (for example, Myrtveit and Stensrud, 1999). Other studies again find CART models superior to regression models (Briand et al. 1999b) whereas other studies report the opposite result (for example Briand et al. 2000). Others again find ANN models superior to regression models (for example, Srinivasan and Fisher, 1995) whereas Jørgensen (1995) reports the opposite result.

So far, nobody has understood why this is so. It has been a puzzle to the research community on software prediction systems for many years. Clearly, we need to consolidate the knowledge on software prediction models; we need to understand why we have

Norwegian School of Management BI
Department of Leadership and Organizational Management/Department of Economics
www.bi.no

obtained so wildly opposing conclusions on this matter.

In this study, we attempt to understand the reason why software researchers obtain so contradictory results. We examine the validation methods as well as the measures used to assess the performance of prediction models. In particular, we investigate how the combined use of cross-validation and the mean magnitude of relative error (MMRE) affect the results.

Using a real data set, we empirically show that MMRE figures may vary wildly, indeed, from zero to large values for almost identical data sets when we evaluate AFA's like EBA, CART and ANN. That is, the MMRE figures are *completely arbitrary* for arbitrary function approximators. Opposed to this, MMRE figures for regression models are consistent across almost identical data sets. The latter result is as we would expect: A small perturbation in the data should yield small differences in MMRE values.

2. TYPES OF COST ESTIMATION MODELS

There are several approaches to cost estimation. One can group them as in Figure 1. Broadly, we may distinguish between sparse-data methods and many-data methods. Sparse-data methods are estimation methods requiring few or no historical data. They include Analytic Hierarchy Procees, AHP, (Shepperd and Cartwright, 2001), expert judgment (Vicinanza et al. 1991) and automated case-based reasoning - CBR (Mukhopadhyay et al. 1992).

Many-data methods may be subdivided into functions and arbitrary function approximators (AFA). Functions are of the general form $y=Ax^B$. Linear regression models, for example COCOMO belong to this class. As opposed to functions, arbitrary function approximators do not make any assumptions regarding the relationship between the predictor and response variables (i.e. between x̲ and y̲). The argument for proposing them is that *"it is very difficult to make valid assumptions about the form of the functional relationship between variables….* [Therefore]… [the] *analysis procedure should avoid assumptions about the relationship between the variables….using more complex functional forms would be difficult since we usually have a poor understanding of the phenomena we are studying."* (Briand et al. 1992). EBA, CART and ANN models belong to the AFA class.

In this paper we only investigate many-data methods.

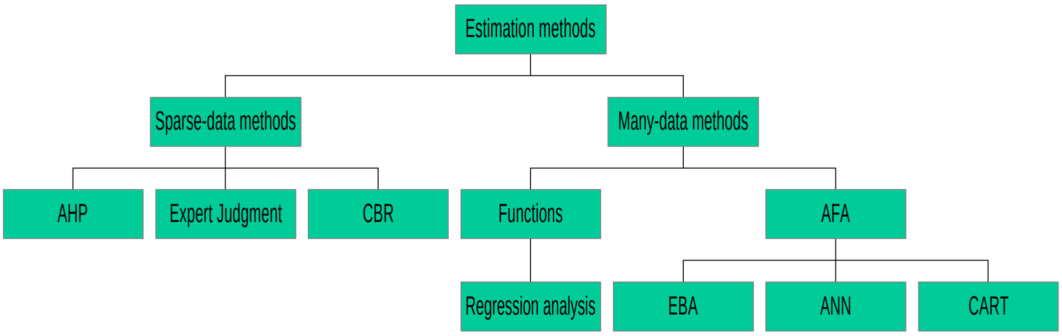

Figure 1. A taxonomy of SW cost estimation methods

3. PREVIOUS WORK ON ESTIMATION METHODS

There exists a relatively large number of empirical studies on software cost estimation models. Especially, there is a large number of studies on regression analysis models since this model often serves as the baseline against which the performance of the other models is compared. See the *Encyclopedia of Software Engineering* (Briand and Wieczorek, 2001) for an overview. It should be observed that e.g. COCOMO is a regression model. Most of the studies have applied the ordinary least squares method. A few studies have also reported on various robust regression methods (Foss et al. 2001; Gray and MacDonell, 1999; Jeffery et al. 2001; Miyazaki et al. 1994; Nesi and Querci 1998; Pickard et al. 1999).

There is also a substantial body of research on AFAs. The latter include CART models (Briand et al. 1998, 1999b; Kitchenham, 1998; Srinivasan and Fisher, 1995), OSR - Optimized Set Reduction, a subtype of CART (Briand et al. 1992, 1993; Jørgensen, 1995), EBA models (Jeffery and Walkerden 1999; Myrtveit and Stensrud, 1999; Shepperd and Kadoda, 2001; Shepperd and Schofield, 1997; Stensrud and Myrtveit, 1998; Walkerden and Jeffery, 1999) and, finally, ANN models (e.g. Samson et al. 1997; Srinivasan and Fisher 1995; Shepperd and Kadoda, 2001).

4. VALIDATION METHODS

Validation methods commonly comprise two elements, the validation procedure and the evaluation criterion (or measure), respectively. Cross validation is a common validation method whereas the mean magnitude of relative error (MMRE) is a common evaluation criterion. We therefore present and discuss both of them.

4.1 Cross-Validation

Cross-validation is a way of obtaining nearly unbiased estimators of prediction error. The method consists of (a) deleting the observations from the data set one at a time; (b) calibrating the model to the *n-1* remaining observations; (c) measure how well the calibrated model predicts the deleted observation; and (d) averaging these predictions over all *n* observations (Efron and Gong, 1983). In software engineering, MRE and MMRE are used as the de facto standard in steps (c) and (d), respectively (Briand and Wieczorek, 2001).

Also, in software engineering, a variant of the cross validation method, v-fold cross validation, is widespread (Briand et al. 1993; Briand et al. 1999a). V-fold cross validation divides the data set into *v* subsets, each with approximately k observations with *k>1*. That is, *v*k□n*. So, rather than deleting one observation at a time, *k* observations are deleted each time. In the machine learning communities within computer science, these subsets are often termed training sets and test sets, respectively.

From a practitioner's standpoint, we think we need to comment on the validity of cross validation vs. the v-fold cross validation assuming a realistic real world situation. What, then, is a real world situation closest to, a normal cross validation or a v-fold cross validation?

To us, it seems that a realistic scenario is as follows. We have a data set with n historical projects, and we are to estimate a single new project. Now, we think it would be wise to use all the n observations to calibrate a model before predicting the effort of the new project. This situation seems perfectly approximated by the normal cross validation procedure where the model is calibrated with *n-1* observations, i.e. one observation less than we would have in the real world case. As opposed to this, the v-fold cross validation removes *k* observations at a time, thereby using a much smaller subset to calibrate the model than would be available in reality.

In this study, we have applied normal cross validation for the AFAs. For the regression analysis, we have not used cross validation at all. We argue that only a small error is introduced in the regression model when we do not remove one observation at a time. There are 38 observations in the data set. Therefore, the impact of a single observation is likely to be small.

4.2 MMRE

The most widely used evaluation criterion to assess the performance of software prediction models is the mean magnitude of relative error (MMRE). This is usually computed following standard evaluation processes such as v-fold cross-validation (Briand and Wieczorek, 2001). Conte et al. (1986) consider *MMRE £ 0.25* as acceptable for effort prediction models. MRE is defined as

$$MRE = \frac{|y - \hat{y}|}{y},$$

where y = actual and $\hat{y}$ = prediction. There exist a number of alleged reasons to use MMRE. It is considered a versatile assessment criterion lending itself to a number of situations. The claimed advantages include the following:

1. Comparisons can be made across data sets (Briand et al. 2000; Shepperd and Kadoda 2001).
2. It is independent of units. Independence of units means that it does not matter whether effort is reported in workhours or workmonths. An MMRE will be, say, 10% whatever unit is used.
3. Comparisons can be made across all kinds of prediction model types (Conte et al. 1986). This means, for example, that it is a valid measure to assess the accuracy of AFA's.
4. It is scale independent. Scale independence means that the expected value of MRE does not vary with size. In other words, an implicit assumption in using MRE as a measure of predictive accuracy is that the error is proportional to the size (effort) of the project (Strike et al., 2001). For example, a 1 person-month error for a 10 person-month project and a 10 person-month error for a 100 person-month will result in equal MREs (10% for both projects).

In this study, we investigate claim 3 and show that cross-validation + MMRE is totally inappropriate to evaluate the performance of an AFA in terms of prediction accuracy.

5. Data used in the study

For the purpose of this study, we use a univariate data set termed the Finnish data set. The predictor variable is function points (FP) and the response variable is effort (development hours). The data set consists of 40 projects (Table 1). Two projects have missing data. The data comes from different companies, and the data collection was performed by a single person. The projects span from 460 to 23000 workhours. Descriptive statistics are provided in Table 1.

Table 1. Descriptive statistics for Finnish data set

Variable	N	Mean	Median	StDev	Min	Max
FP	40	761	638	511	65	1814
Effort	38	7573	5430	6872	460	23000

In Figure 2, we have plotted effort against FP where we observe that the data are heteroscedastic (increasing variance). This data set is termed the *original* data set.

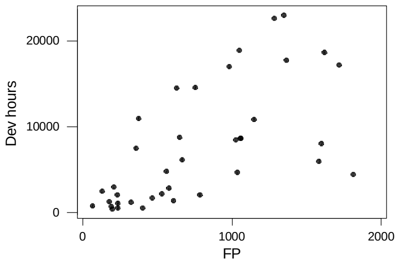

Figure 2. Plot of effort vs. size, original Finnish data set.

For the purpose of this study, we have made a slight modification to the original Finnish data. In Figure 3, we have plotted the data after the modification. We have paid care not to change the fundamental characteristics of the data such as variance, heteroscedasticity, range, and number of observations (still 38).

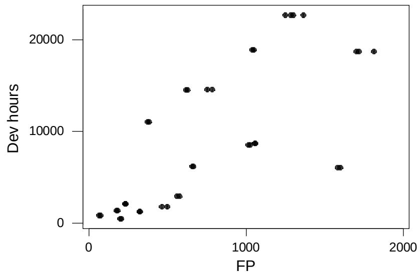

Figure 3. Plot of effort vs. size, modified Finnish data set.

To understand what we have done to the original data, we zoom in on the observations in Figure 4. The perturbation consists of identifying pairs of projects that are close to each other and modifying these so that they come slightly closer to each other. Specifically, we have modified the effort so that the effort of such a pair of observations is identical. Furthermore, we have made slight modifications in the FP dimension to ensure that e.g. B is closer to A than to C in the FP dimension. The modification therefore rearranges the observations so they are similar to the pattern in Figure 4.

We would expect such small perturbations in the data to result in small changes in the models fitted to the data and consequently, small changes in the results in terms of MMRE. In the study, we show that only functions like regression analysis models behave as expected. Small changes in the data result in small changes in the regression model as well as in small differences in MMRE. As opposed to functions, the MMRE figures of AFA models are extremely sensitive to these perturbations.

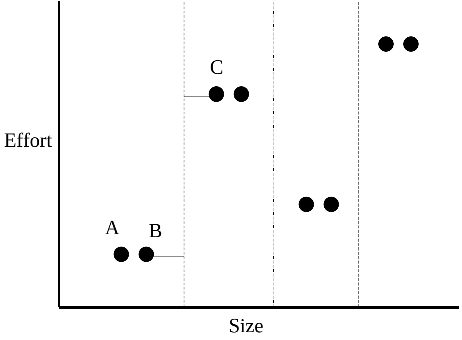

Figure 4. Zoom in of how Finnish data modification has been performed

6. LINEAR REGRESSION MODELS

Linear regression analysis comprises a family of techniques for fitting a line (in the univariate case) to a set of observations. Thus, it provides us with an equation describing the relationship between FP and effort. In case the data points exhibit non-linear effects, this is discovered by error analysis, analysis of residuals. If such effects are found, the scale may be transformed so the transformed data exhibit linear characteristics. In this way, a linear regression model may still be used for a non-linear data set. For the Finnish data set, we observe that a log-log transformation seems satisfactory. Performing this transformation, the data comply with the assumptions of OLS regression. In particular, the relationship between FP and effort seem reasonably linear; the residuals are normal (not reported); the data are reasonably homoscedastic.

The model calibrated on the original data is given in Table 2. The model calibrated on the modified data is given in Table 3. The OLS regression model for the original and modified data sets are reasonably similar, as we would expect for relatively small perturbations of the data. From Table 2 and Table 3, we observe that the coefficients are similar (Coef); the standard error of the coefficients is similar (SE coef and T); and the goodness of fit metric is similar (R^2).

The point estimate from an OLS regression model is the expected value or *mean*. That is, the point estimate is a well-defined statistic. The expected value is the *most likely* value of the actual effort. One desirable property of the mean is that the probability of exceeding it is 50%. In addition, the OLS regression model supplies *prediction intervals* (95% PI; dashed lines in Figure 5). In this case, we have shown the 95% prediction intervals. That means there is a 5% chance of exceeding the upper bound of the prediction interval. For example, suppose we need to predict the effort of a 1026 FP project (ln(1026 FP)=6.93). Then we obtain mean_effort = 8,100 workhours, and lower/upper bounds = 1,591/41,237 workhours for the 95% prediction interval. This information advices the customer to budget for at least 40,000 workhours if he is risk-averse, but that the most likely outcome is 8100 work hours. In other words, the worst case may exceed the expected value by a factor of five based on this particular historical data set. Also, knowing that there is a large uncertainty of the 8100 workhours expected value, the customer is better positioned to make an informed decision of buy/not buy.

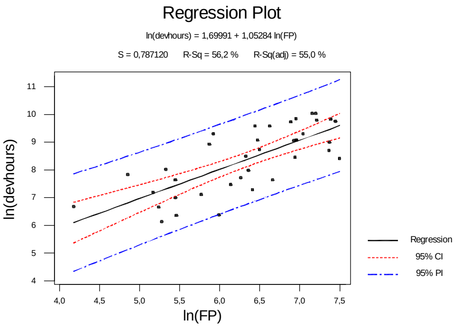

Figure 5. Log-log linear regression model with 95% prediction interval (95% PI lines), Finnish data set.

Table 2. Log-log regression model, original Finnish data set.

Predictor	Coef	SE coef	T
const	1.70	0.99	1.71
ln(FP)	1.05	0.16	6.8

R^2	0.56

Table 3. Log-log regression model, modified Finnish data set.

Predictor	Coef	SE coef	T
const	1.12	0.83	1.35
ln(FP)	1.17	0.13	9.1

R^2	0.70

7. AFA MODELS

7.1 EBA

EBA methods identify analogues (or similar cases) in the database. Commonly used similarity measures are Euclidean distance and correlation coefficients. Euclidean distance is employed in the EBA tool ANGEL (Shepperd and Schofield, 1997). ANGEL predicts effort based on identifying analogous or *similar* projects for which effort is known. The predicted effort is basically identical to the effort of the most similar project. The ANGEL model is illustrated in Figure 6.

Using the Finnish data set (section 5) as example of how ANGEL works, the most similar project is the project which is closest in terms of FP (in the univariate case). For example, to estimate the effort for a 1300 FP project, we would measure the distance to every project in the database and identify project C (1282 FP) in Figure 6 as closest (a distance of 18 FP). C is closer than for example D (1347 FP). The effort for C is 22670 workhours. Therefore, the estimated effort for the 1300 FP project would be 22670 workhours.

We observe that the similarity measurements may be used to *rank* all the projects in the database with respect to closeness with project X where X is 1300 FP in our particular example.

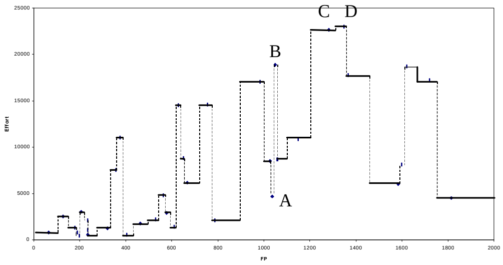

Figure 6. Estimation by analogy model (ANGEL), Finnish data set

ANGEL may also compute estimates that are averages of the *n* closest projects where *n* may be any value chosen by the user. If we are extreme, we may average over all the 38 projects in the Finnish data set. In this case, the ANGEL function would look like the solid thick line in Figure 7. (The average effort of all projects is 7573 workhours.) For any other *n*, the model would be a stepwise function somewhere in between the solid thick line and the collection of the thin horizontal, discontinuous line segments in Figure 7.

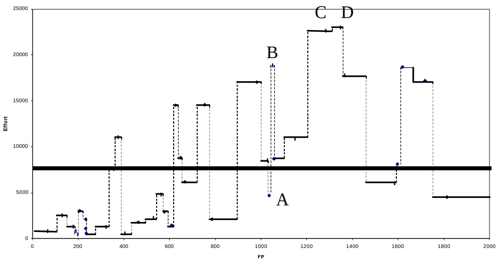

Figure 7. ANGEL prediction model taking the average over the 38 closest projects, Finnish data set (thick solid line).

7.2 CART

Decision tree approaches classify the data set in a tree structure (Brieman et al. 1984). Decision trees are referred to as classification or regression trees depending on whether they classify discrete variables or continuous variables, respectively. The common term for these trees is CART, Classification And Regression Trees. For the Finnish data set, both the predictor variable (FP) and the response variable (effort) are continuous. In this case, we therefore use a *regression* tree (RT).

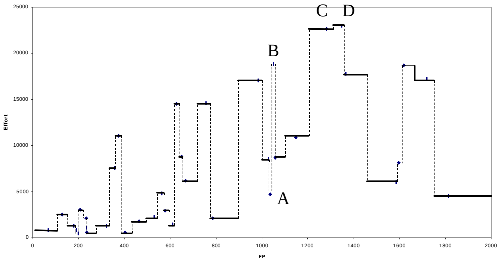

Figure 8. CART model, Finnish data set.

The idea is to partition projects into more homogeneous subsets based on the *similarities* of projects *within* groups and dissimilarities between groups. There are many ways to operationalize "similarity". Srinivasan and Fisher (1995) used the *minimum mean squared error* (MSE) of the response variable as the criterion to divide a group into two disjoint subgroups. Suppose further that we adopt Srinivasan and Fisher's strategy, which was to divide the data set into "maximum depth". *"We allowed the regression tree to grow to a "maximum depth", where each leaf represented a single software project…(p.130)"* For the Finnish data set, we then obtain 38 leafs, and the mean effort per leaf equals the actual effort for the single project in that leaf. This CART model is illustrated in Figure 8.

To estimate a new project using CART, we must find out which leaf it is closest to in the FP dimension and then use the actual effort of the closest leaf as the estimate. Using Euclidean distance to identify the closest leaf, we obtain *exactly the same* function and results as for ANGEL (Compare Figure 6 and Figure 8). The CART model in Figure 8 is identical to the EBA model in ANGEL. That is, ANGEL and CART gives us identical estimates. Thus, the critique of ANGEL applies equally well to CART given that

- there is one single predictor variable
- CART is allowed to grow to leafs of one project
- ANGEL uses the single closest analogy

Of course, both CART and ANGEL would give somewhat different results if we apply some "filtering". Filtering in ANGEL means that we use the mean of the *n* closest analogies as an estimate rather than the single closest analogue (with n>1).

Filtering in CART means that we do not let the tree grow to maximum depth (i.e. n>1 in each leaf). If we let CART grow to "minimum depth", we get a function that is a solid horizontal line just like for ANGEL as shown in Figure 7.

Whatever the value of *n*, we argue that the fundamental structure and properties remain the same for both CART and ANGEL. The basic nature of a CART or an EBA model like ANGEL, regardless of *n*, is an "up-and-down staircase" function without any well-defined properties of the estimates and with a possibility of obtaining *lower* estimates as size (FP) increases. This seems contrary to common sense.

7.3 ANN

The vision of artificial intelligence (AI) research is to devise systems that behave like intelligent, living creatures that can learn from experience and modify their behavior accordingly. Artificial neural network (ANN) models are inspired by biological neural networks.

ANN models employ a wide variety of algorithmic approaches and architectures. A main concern in computer science has been to devise algorithms that are computationally efficient and require small memory space. One of the merits of ANN models from a computational perspective is that they lend themselves to parallel processing. Thus, they are suited to multi-CPU hardware architectures.

Whatever the algorithm used, the output from an ANN model is a smooth curve through the observations (Srinivasan and Fisher, 1995). An ANN model is therefore a *smooth* AFA rather than a *stepwise* AFA (as CART and EBA are). An ANN model may be fitted more or less to the observations through more or less filtering. The number of nodes determines the degree of fit. In Figure 9, we have shown an ANN model with high fit (many nodes).

Just like for CART and EBA models, an ANN model has a possibility of giving *lower* estimates as size (FP) increases. Again, this seems contrary to common sense.

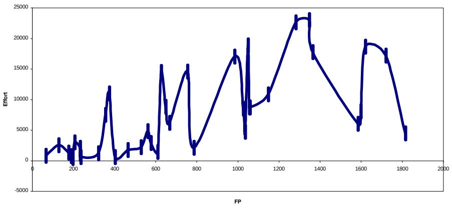

Figure 9. ANN model, Finnish data set.

8. Results

In this section, we report the results in terms of MMRE of regression models, EBA and CART models on the original data and the modified data, respectively. The results in terms of MMRE in Table 4 clearly demonstrate that an AFA like ANGEL or CART is extremely sensitive to small perturbations of the data whereas a function like a regression model is not. The MMRE for the two regression models does not change by more than 25% (R column).

We have not reported MMRE numbers for ANN because we did not have an appropriate tool to calculate this. However, we have good reasons to believe that the results in terms of MMRE would be arbitrary, too, depending on details in the pattern of observations that ought not to influence on a model performance criterion.

Table 4. MMRE of AFA (ANGEL/CART) and regression model (R).

Data	N	MMRE (AFA)	MMRE (R)
Finnish, orig.	38	1.55	0.79
Finnish, modif.	38	0.00	0.63

As opposed to the MMRE results for the regression model, we observe that the MMRE results for the AFA models (CART/ANGEL) are completely different for the original data and the modified data; MMRE varies by *infinitely* many percent (from 0.00 to 1.55). That is, for two rather similar data sets, the AFA models seemingly perform from anywhere between extremely well to very bad.

Thus, if we had compared a regression model with an AFA on a data set like the original Finnish data set, the regression model would have obtained the lowest MMRE and been deemed best. Comparing the same two models on a slightly different data set like the modified data set, the AFA would apparently have outperformed the regression model completely in terms of MMRE and been hailed as the ultimate, perfect model.

9. Conclusions

In this study, we have contributed an explanation of why studies on prediction models do not converge when AFA's are part of the study. Specifically, we have shown that the de facto evaluation method, cross-validation combined with MMRE, yields completely *arbitrary* MMRE values for arbitrary function approximators.

For data sets that are almost identical in terms of properties like variance, heteroscedasticity, linearity, range, number of observations, etc., we may, when evaluating and AFA, obtain wildly different prediction accuracies in terms of

MMRE. We have shown that we may obtain anything from a spectacular performance (e.g. MMRE=0) to very low performance (i.e. a large MMRE) in terms of MMRE.

The implications for previous research on AFA-type prediction models are that the conclusions of these studies with regard to the research question: "which model is best?" are of limited, or probably no, value.

Obviously, other validation methods are called for to properly assess AFA's.

10. References

LC Briand, VR Basili, and WM Thomas (1992), "A Pattern Recognition Approach for Software Engineering Data Analysis", *IEEE Trans. Software Eng*., vol. 18, no. 11, pp. 931-942.

LC Briand, VR Basili, and CJ Hetmanski (1993), "Developing Interpretable Models with Optimized Set Reduction for Identifying High-Risk Software Components", *IEEE Trans. Software Eng.*, vol. 19, no. 11, pp. 1028-1044.

LC Briand, K El-Emam, and I Wieczorek (1998), "A Case Study in Productivity Benchmarking: Methods and Lessons Learned", *Proc. 9th European Software Control and Metrics Conference* (*ESCOM)*, Shaker Publishing BV, The Netherlands pp. 4-14.

LC Briand, K El-Emam, and I Wieczorek (1999a), "Explaining the Cost of European Space and Military Projects", *Proc. 21st International Conference on Software Engineering* (ICSE 21), ACM Press, pp. 303-312.

LC Briand, K El-Emam, K Maxwell, D Surmann, and I Wieczorek (1999b), "An Assessment and Comparison of Common Cost Software Project Estimation Methods", *Proc. 21st International Conference on Software Engineering* (ICSE 21), ACM Press, pp. 313-322.

LC Briand, T Langley, and I Wieczorek (2000), "A replicated Assessment and Comparison of Common Software Cost Modeling Techniques", *Proc. International Conference on Software Engineering*, (ICSE 22), ACM Press, pp. 377-386.

LC Briand and I Wieczorek (2001), "Resource Modeling in Software Engineering", in *Encyclopedia of Software Engineering*, 2 Volume Set, (Ed. J. Marciniak) Wiley.

Brieman L, Friedman J, Olshen R, and Stone C, Classification and Regression Trees, (Wadsworth Inc., Belmont CA, 1984).

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

SD Conte, HE Dunsmore, and VY Shen (1986), *Software Engineering Metrics and Models*, Benjamin/ Cummings, Menlo Park CA.

B Efron, and G Gong (1983), A Leisurely Look at the Bootstrap, the Jackknife, and Cross-Validation, *The American Statistician*, February, Vol. 37, Issue 1, pp. 36-48.

T Foss, I Myrtveit, and E Stensrud (2001), "A Comparison of LAD and OLS Regression for Effort Prediction of Software Projects", *Proc. 12th European Software Control and Metrics Conference* (*ESCOM 2001)*, Shaker Publishing BV, The Netherlands, pp. 9-15.

AR Gray, and SG MacDonell (1999), "Software Metrics Data Analysis - Exploring the Relative Performance of Some Commonly Used Modeling Techniques", *Empirical Software Engineering*, 4, pp.297-316.

R Jeffery and F Walkerden (1999), "Analogy, Regression and Other Methods for Estimating Effort and Software Quality Attributes", *Proc. ESCOM'99*, Shaker Publishing BV, The Netherlands, pp. 37-46.

R Jeffery, M Ruhe, and I Wieczorek (2001), "Using Public Domain Metrics to Estimate Software Development Effort", *Proc. METRICS 2001*, IEEE Computer Society, Los Alamitos CA, pp. 16-27.

M Jørgensen M (1995), "Experience With the Accuracy of Software Maintenance Task Effort Prediction Models", *IEEE Trans. Software Eng.* 21, 8, pp. 674-681.

BA Kitchenham (1998), "A Procedure for Analyzing Unbalanced Datasets", *IEEE Trans. Software Eng.*, vol. 24, no. 4, pp. 278-301.

Y Miyazaki, M Terakado, K Ozaki, and H Nozaki (1994), "Robust Regression for Developing Software Estimation Models", *Journal of Systems and Software*, Vol. 27, pp. 3-16.

T Mukhopadhyay, SS Vicinanza, and MJ Prietula (1992), "Examining the Feasibility of a Case-Based Reasoning Model for Software Effort Estimation", *MIS Quarterly*, June, pp. 155-171.

I Myrtveit, and E Stensrud (1999), "A Controlled Experiment to Assess the Benefits of Estimating with Analogy and Regression Models", *IEEE Trans. Software Eng.* 25, 4, pp. 510-525.

P Nesi and T Querci (1998), "Effort Estimation and Prediction for Object-Oriented Systems", *Journal of Systems and Software*, vol. 42, pp. 89-102.

L Pickard, B Kitchenham, and S Linkman (1999), “An Investigation of Analysis Techniques for Software Datasets”, *Proc. METRICS 99*, IEEE Computer Society, Los Alamitos CA, pp. 130-142.

B Samson, D Ellison, and P Dugard (1997), “Software Cost Estimation Using and Albus Perceptron (CMAC)”, *Inf. and Software Tech.* 39, pp. 55-60.

M Shepperd and C Schofield (1997), “Estimating Software Project Effort Using Analogies”, *IEEE Trans. Software Eng*., vol. 23, no. 12, pp. 736-743.

M Shepperd and M Cartwright (2001), Predicting with Sparse Data, *IEEE Trans. Software Eng.*, vol. 27, no. 11, pp. 987-998.

M Shepperd and G Kadoda (2001), “Comparing Software Prediction Techniques Using Simulation”, *IEEE Trans. Software Eng*., vol. 27, no. 11, pp. 1014-1022.

R Srinivasan and D Fisher (1995), “Machine Learning Approaches to Estimating Software Development Effort”, *IEEE Trans. Software Eng*., vol. 21, no. 2, pp. 126-137.

E Stensrud and I Myrtveit (1998), “Human Performance Estimating with Analogy and Regression Models: An Empirical Validation”, *Proc. METRICS’98*, IEEE Computer Society Press, Los Alamitos CA, pp. 205-213.

K Strike, K El-Emam, and N Madhavji (2001), “Software Cost Estimation with Incomplete Data”, *IEEE Trans. Software Eng*., vol. 27, no. 10, pp. 890-908.

SS Vicinanza, T Mukhopadhyay, and MJ Prietula (1991), Software Effort Estimation: An Exploratory Study of Expert Performance, *IS Research*, 2:4, pp. 243-262.

F Walkerden and R Jeffery (1999), “An Empirical Study of Analogy-based Software Effort Estimation”, *Empirical Software Eng.*, 4, 2, pp. 135-158.