ОЦІНКА ВИТРАТ ТА ВАРТОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

UDC 004.413:338.5(045)

D. Batsenko
National Aviation University

METHOD OF CALIBRATION OF COCOMO MODEL VIA REDUCTION OF THE MAIN EQUATION

This article reviews method of calibration of COCOMO software cost estimation model by reduction of the main equation as well as scientific and mathematical method that lied foundations for it.

У статті розглядається метод калібрування моделей оцінки вартості програмного забезпечення COCOMO шляхом редукції основного рівняння і наукові та математичні методи, що були покладені в його основу.

В статье рассматривается метод калибровки модели оценки стоимости программного обеспечения COCOMO путем редукции основного уравнения, а также научные и математические методы, которые легли в его основу.

Keywords: calibration, COCOMO, software.

Introduction

With the significant growth of software complexity methods for software cost estimation became necessary condition of the success of any software project. But over the years of software cost estimation models' improvement still most of the models are not generalized and that is the reason for the appearance of various calibration techniques and methods aiming to improve the quality of software cost estimation results of a given model for a specific company domain.

This article presents the results of scientific research in the field of software cost estimation model calibration and proposes the special method of calibration of COCOMO model via reduction of the main equation of the model.

The mathematical model

In this method, a number of ideas are taken from the relevance of features that were discussed in [1], and the evaluation criteria for prediction models in [2], [3] and [4]. This method aims to find the optimal feature subset that enables higher accuracy and lower variability of results than the general model with the full feature set. Therefore it is important to build the mathematical model and define corresponding terminology. For example, the relevance of features is defined to show whether the feature subset is relevant to the model or not. The optimal feature subset not always includes all relevant feature subsets but generally it shouldn't include the irrelevant feature subset.

$$PM = a * \left(KSLOC^{b}\right) * \left(\prod EM_j\right) \quad \text{(Eq. 1)}$$

where

PM – person months;

EM – effort Multipliers shown in Table2.4;

KSLOC – size as thousand lines of code, is estimated or converted from a function point metric;

a and b – domain-specific parameters/constants.

$$PM = A * \left(KSLOC^{B+1.01*\sum_{i=1}^{5} SF_i}\right) * \left(\prod_{j=1}^{17} EM_j\right) \text{ (Eq. 2)}$$

where

A – baseline multiplicative constant;

B – baseline exponential constant;

Size – Size of the software project measured in terms of KSLOC (thousand of source lines of code) or function points related to programming language;

SF – scale factor;

EM – effort multiplier;

Definition 1. Model. These models are the same as the ordinary COCOMO 81 model shown in Equation 1 or the COCOMO II model shown in equation 2 except that it uses fewer model parameters (calibration features, e.g. effort multipliers).

Definition 2. Feature. A feature, sometimes called a parameter, an attribute, a factor, or a cost driver, describes some characteristics of a project instance.

Definition 3. Feature Subset. A feature subset includes one or more than one but not all parameters of the model.

Definition 4. Full Feature Set. A full feature set includes all parameters of the model.

Definition 5. Accuracy. COCOMO's performance is often measured in terms of PRED(30). PRED(N) is calculated from the relative error, or RE (shown in equation 3), which is the relative size of the difference between the actual and estimated value. Given a data set of size D, a Training set of size (Train=|Train|) ≤ D, and a test set of size T=D- |Train|, then the mean magnitude of the relative error, or MMRE (the mean magnitude of relative error, shown in equation 5), is the percentage of the absolute values of the relative errors, or MRE (the magnitude of relative error shown in equation 4), averaged over the T items in the test set. PRED(N) for each hold-out experiment is calculated with equation 6.

$$RE_i = \frac{estimate_{\ i} - actual_{\ i}}{actual_{\ i}} \qquad \text{(Eq. 3)}$$

$$MRE_i = abc(RE_i) \qquad \text{(Eq. 4)}$$

$$MMRE = \frac{100}{T}\sum_{i=1}^{T} MRE_i \qquad \text{(Eq. 5)}$$

$$PRED(N)_h = \frac{100}{T}\sum_{i=1}^{T}\begin{cases} 1 \ if \ MRE_i \le \frac{N}{100} \\ 0 \ otherwise \end{cases} \qquad \text{(Eq. 6)}$$

In this approach, Hold-out experiments are conducted; the accuracy of the model is defined in equation 7 as the mean of PRED(N) in all hold-out experiments in the same calibration dataset:

$$PRED(N) = \frac{1}{n}\sum_{i=1}^{n} PRED(N)_h^i \qquad \text{(Eq. 7)}$$

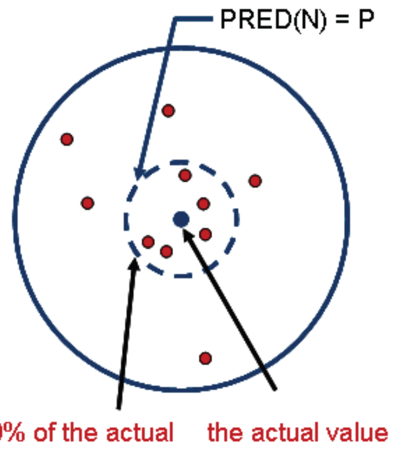

Figure 1. An Example for PRED(30)=50

An example is shown in Figure 1, a PRED(30)=50% means that half the estimates are within 30% of the actual results. The results are reported in terms of PRED(N), not MMRE. This is a pragmatic decision as PRED(N) is easier understood by business users than MMRE. Also, there are more PRED(N) in reports in the literature than MMRE, possibly due to the influence of the COCOMO researchers who reported their 1999 study using PRED(N) [5].

Definition 6. Variability. PRED(N) is calculated for different "holdout" samplings of the calibration data. Holdout samplings use randomized subsamples of the data to calibrate PRED value and the unsampled data to calculate PRED value. Different samplings produce different PRED(N) values. M shown in equation 8 is denoted as the mean of PRED(N) from all hold-out experiments. Variability of the estimation in the model, denoted as *V* in equation 9, shows how much spread is in the estimation. Standard deviation σ in Equation 10 is used to measure the variability of the accuracy of the model in this approach:

$$\mu(x) = \frac{1}{N}\sum_{i=1}^{N} x_i \qquad \text{(Eq. 8)}$$

$$\mu(x) = \frac{1}{N}\sum_{i=1}^{N} x_i \qquad \text{(Eq. 9)}$$

$$\sigma = \sqrt{V} \qquad \text{(Eq. 10)}$$

A smaller σ provides more "confidence" in using PRED(N)=μ than larger σ since all the values are closer to μ . Small σ indicates small variability in estimations while large σ indicates large variability in estimation.

Definition 7. Better feature subset. Given an learner L, a training dataset and a test dataset with the feature subsets X_1 , X_2 , ..., X_n , a better feature subset, X_{bet}, is the feature subset X^i that provides higher accuracy without increasing variability than those of the full feature set X of the general the model:

Accuracy(X^i) > *Accuracy*(X) i *Variability*(X^i)<*Variability*(X)

For any $1 \le j \le n$ (Eq. 12)

Definition 8. The best accuracy feature subset. Given a learner L, a training dataset and a test set with the better feature subsets X^1_{bet}, X^2_{bet} , ..., X^n_{bet} , the best accuracy feature subset, X_{acc}, is the better subset X^i_{bet} that maximizes the accuracy among the better feature subsets:

$Accuracy(X^i_{bet})Accuracy(X^j_{bet})$For any $1 \le j \le n$ (Eq. 13)

Definition 9. The least variability feature subset. Given a learner L, a training dataset and a test set with the better feature subsets X^1_{bet}, X^2_{bet} , ..., X^n_{bet} , the least variability feature subset, X_{sd}, is the better subset X^i_{bet} that minimizes the SD (standard deviation) among the better feature subsets:

$$SD(X^i_{bet}) \le SD(X^j_{bet})$$

For any $1 \le j \le n$ (Eq. 14)

Definition 10. The optimal feature subset. Given a learner L, a training dataset and a test set with the better feature subsets X^1_{bet}, X^2_{bet}, ..., X^n_{bet}, an optimal feature subset, X_{opt}, is the better subset X^i_{bet} that maximizes the accuracy among the better feature subsets:

$$Accuracy\left(X^i_{bet}\right)Accuracy\left(X^j_{bet}\right)$$

For any $1 \le j \le n$ (Eq. 15)

Definition 11. The relevance of a feature. If adding a feature X_i into any feature subset that does not include X_i , or removing X_i from any feature subset that includes X_i will change the accuracy of the model, the feature X_i is strongly relevant to the model. If a feature X_i is not strongly relevant and there exists such a feature subset, adding X_i into the feature subset that does not include X_i or removing X_i from the feature subset that

includes X_i will change the accuracy of the model, the feature X_i is weakly relevant to the model. If a feature X_i is strongly relevant or weakly relevant, X_i is relevant to the model. If a feature X_i is neither strongly relevant nor weakly relevant, X_i is irrelevant to the model. This approach apply the Wrapper – feature subset selection implementation method with k-fold cross validation to evaluate the features of the model and N is the number of how many times a feature X_i is selected in the k-fold cross validation experiments:

1) X_i is strongly relevant if and only if $N = k$ (such as Size);

2) X_i is weakly relevant if $1 \le N < k$;

3) X_i is relevant if $1 \le N \le k$;

4) X_i is irrelevant if and only if $N = 0$. (Eq. 16)

Normally, removing the irrelevant features results in improvement of the performance of the model. The strong relevant features should always be kept in the model and removing any strong feature will degrade the performance of the model.

The next section will show how this model is implemented with the machine learning methods and statistical methods.

Methods Applied in the Approach. Machine Learning is defined as the study of computer algorithms that improves automatically through experience [6]. Applications with machine learning techniques learn when they change their behavior in a way that makes them perform better in the future [7]. A number of research applied machine learning and statistical methods in software cost estimation [8], [9], [10], [11], [5], [12], [13], [14], and [15]. The most successful approach is [5], which has been used to calibrate COCOMO II from 1998. In this research approach shown in Figure 2 for software cost estimation, machine learning techniques are used to formulate the process and build the model from the training data, and statistical methods are used to test, validate and evaluate the process and the model built by machine learning on the test set.

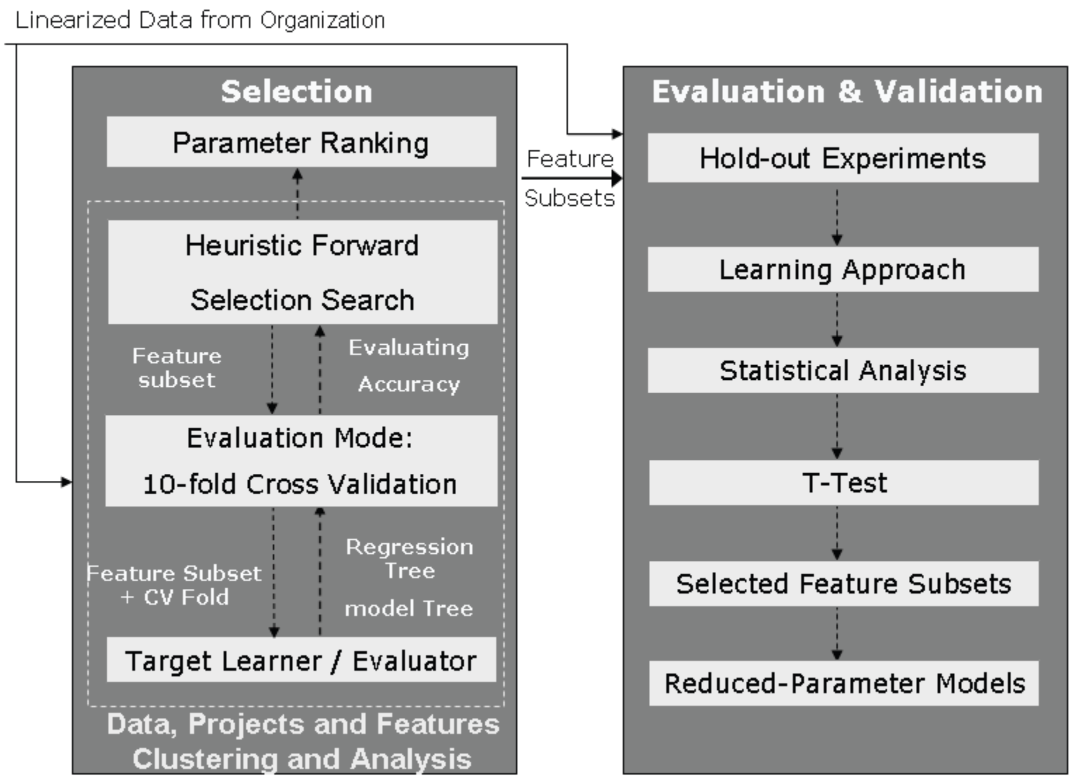

Figure 2. The integration of different techniques and methods in the approach

Linearization

Ordinary least squares regression and M5 model tree are linear models. It is known that in case when linear model is applied to a non-linear relationship, the performance of the model will be decreased. COCOMO 81 shown in Equation 17 and COCOMO II shown in Equation 18 are exponential models, with the assumption that the changes of effort valu grow faster than the changes of size value. The logarithmic transformation were used to transform COCOMO 81 and COCOMO II into linear models.

Linearized COCOMO 81 model:

$$ln(PM) = \beta_0 + \beta_1 * \ln(Size) + \beta_2 * \ln(EM_2) + \cdots + \beta_{16} * \ln(EM_{16}) \quad \text{(Eq.17)}$$

Linearized COCOMO II model:

$$ln(PM) = \beta_0 + \beta_1 * \ln(Size) + \beta_2 * 0.01 * SF_1 * \ln(EM_2) + \cdots + \beta_6 * 0.01 * SF_5 * \ln(Size) + \beta_7 * \ln(EM_1) + \cdots + \beta_{23} * \ln(EM_{17}) \quad \text{(Eq. 18)}$$

All these transformations create a new parameters that are full mathematical equivalents to the original parameters, but are expressed in different measurement.

Clustering and Analyzing Project Features

In this approach, the most promising features in a given dataset are identified with learning algorithm - FSS (Feature subset selection). As the data sets may contain several extraneous features which can reduce the efficiency of the model, this approach helps us identify the important attributes and remove redundant ones.

If only the most relevant features were to be selected and given to the learning algorithm they can produce smaller models. This enhances the understanding of the dataset or domain under consideration. Dimensionality reduction also speeds up the learning process.

In this study, the WRAPPER FSS method is applied, which is a FSS method evaluating parameter sets by using a learning scheme and statistical re-sampling technique such as cross validation to estimate the accuracy of the learning scheme for a set of attributes, and implemented in WEKA [7] (which is a data mining toolkit, free, open source, well documented, compatible on many platforms, and easy to install). When using WRAPPER, a target learner is augmented with a preprocessor that used a heuristic forward select search to grow subsets of the available features. At each step in the growth, the target learner is called to determine the performance of the model learned from the current subset. Subset growth is stopped when the growth is stale; i.e. after a MAX STALE number of times, adding attributes has not improved the performance.

For example, suppose the set of attributes were {A,B,C,D,E,F,...,Z} and MAX STALE was 2. WRAPPER starts by selecting one attribute at random (e.g. C) and score its performance.

Selected = {C} Score = 30 Stale = 0

Next, another randomly selected attribute (e.g. B) is added and scored:

Selected = {C,B} Score = 50 Stale = 0

Note that the addition of B is not a stale addition since it improves the score.

However, the addition of the next randomly selected attribute (e.g. E) does not improve the score, thus is stale increments:

Selected = {C,B,E} Score = 40 Stale = 1

Similarly, adding D also fails to improve the score beyond just using {C,B} thus is scale increments again.

Selected = {C,B,E,D} Score = 42 Stale = 2

Since MAX STALE has been reached, WRAPPER would remove from the selected set all the attributes implicated in the stale growth ({E,D}). The search would then continue, using other attributes.

Figure 3 shows Wrapper is applied in RPM approach. First, the data is divided into 10 equal-size subset randomly. For each time, one leave-one-out data is used. The feature list - FL is the input of the best first search. In the first round, the search sends each feature in the feature list to the target learner, then the target learner learn the model with that feature plus the features from the selected list - SL, and the evaluation function is applied. After the first round, the search adds the feature that makes the model perform best into the selected list and remove it from the feature list. Then it checks whether it should stop or not. Then the following is the second round, the third round … and so on.

In RPM, minimizing the root mean-square error (RMSE) is used as the evaluation function. The “Stop criteria” is that if the RMSE is increased each time in the last 5 expansions, the search stops. If it does reach the stop criteria, it output the selected feature subset that yields the best performance in the model.

There are 10 leave-one-out data, so 10 selected feature subsets are obtained.

Then RPM uses how often the feature is selected as its ranking. Note that, SIZE is always selected and it gets the highest ranking.

The core technology used in this study is FSS (feature subset selection) and FSS is an efficient heuristic search through subsets of the available attributes. The goal of this search is to find a subset that gives similar, if not superior, performance than using all the attributes. Equation 1 demonstrates how large that space can be.

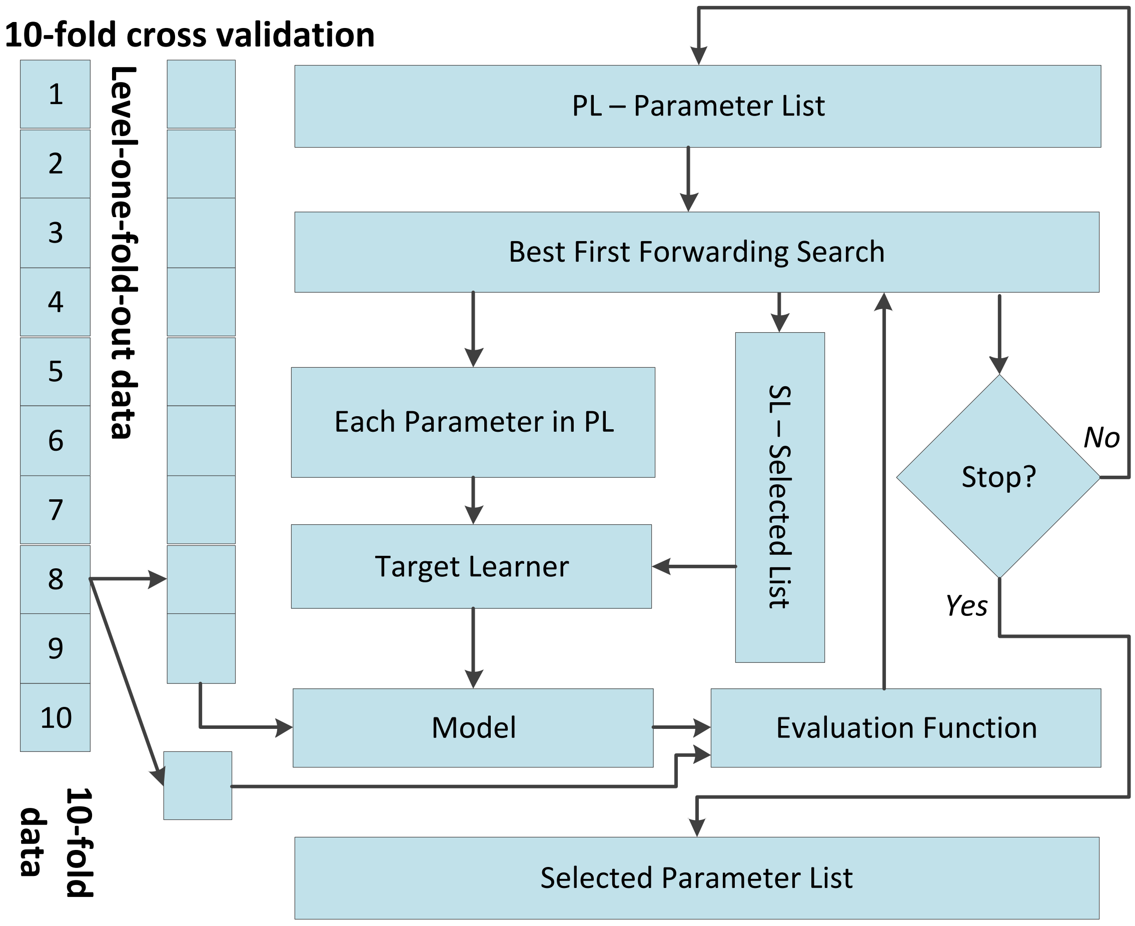

Figure 3. The Wrapper in RPM

There are 15 parameters except for SIZE (total 16 parameters in COCOMO 81). An exhaustive search through all possible subsets would have to explore the 32768 sets as shown in Equation 19. It is assumed that only 6 most promising feature subsets are identified with the method, and 20 seconds are needed for each hold-out experiment (training set and test set are separated) for each PRED of PRED(25, 30) on 60 project instances in that domain, the number of total seconds as shown in equation 20, indicates that 1.25 years are needed to build the model shown in equation 21.

$$FeatureSubset = \sum_{i=1}^{15} C_{15}^{i} = 32768 \quad \text{(Equation 19)}$$

$$Holdout = 0.5 * 30 * 32768 * 2 = 983040 \quad \text{(Eq. 20)}$$

$$Day = 983040/60/60/24 = 11.38 \quad \text{(Eq. 21)}$$

$$Seconds = 0.5 * 30 * 6 * 2 = 18 \quad \text{(Eq. 22)}$$

Fortunately, this study did not take 11.38 days. The FSS methods used in this research is so efficient that these experiments required only the 180 seconds as shown in Eq. 22.

One of the major advantages of the WRAPPER approach is that, if some target learner is already implemented, then the WRAPPER is simple to implement. Also, in their comparative evaluation of feature subset selection techniques [16], Hall and Holmes conclude that WRAPPER is the best FSS mechanism, if the data set is not too large. At each step in the heuristic search, WRAPPER makes another call to the target learner. Hence, it may be too slow for large data sets. The data sets used in this study are small (maximum size: 200 instances) and hence are amenable for WRAPPER.

Table 1
Wrapper Results for the COCOMO 81 Data

COCOMO 81 data with LSR Approach: 63 Instances						
Parame-	60 Folds	50	40	30	20	10 Folds
VEXP	100	100	100	100	100	100
LEXP	100	98	100	100	100	100
TIME	95	98	93	100	100	100
LOC	100	100	100	100	100	100
RELY	93	98	93	100	100	90
PCAP	72	82	73	63	70	60
AEXP	82	76	68	73	70	50
TURN	80	86	78	63	75	50
ACAP	47	48	50	50	55	50
SCED	75	80	73	53	75	40
DATA	15	12	18	13	10	30
STOR	15	12	18	13	10	30
MODP	18	24	25	20	20	20
CPLX	15	12	25	23	20	20
VIRT	12	8	18	13	10	10
TOOL	13	6	8	3	25	0

Table 2
Wrapper Results for the NASA 60 Project Data

NASA 60 data with LSR Approach: 60 Instances						
Parame-	60	50	40	30	20	10 Folds
TURN	100	98	100	100	100	100
ACAP	100	100	100	97	100	100
TIME	100	100	100	100	100	100
LOC	100	100	100	100	100	100
STOR	95	92	93	87	90	80
VEXP	60	60	65	67	70	70
DATA	23	14	20	20	20	40
AEXP	8	2	5	3	0	20
PCAP	5	10	5	7	0	10
MODP	0	14	3	3	5	10
VIRT	0	0	8	3	20	10
RELY	3	2	5	7	0	0
TOOL	0	0	0	3	10	0
CPLX	3	10	5	7	5	0
LEXP	2	10	0	0	5	0
SCED	5	4	3	3	0	0

Wrapper uses k-fold cross validation in Wrapper. It is applied with LSR approach on COCOMO 81 and NASA 60 project data from 10 folds to 60 folds to chose the right number of folds needed. As shown in Table 2 and 3.2, there is no evidence that conducting more than 10-way cross values alters the conclusions that could be found in a 10-fold. A similar pattern was observed by

Wrapper with another approach M5 model tree. There is another interesting result: COCOMO81 data are much more diverse than NASA data as the significance of different parameters in COCOMO 81 are more effective than those in NASA. The parameters are more correlated in the projects that could have similar nature from an organization than in those projects crossing different organizations. Removing such correlated features will increase the performance of the model.

Best First forward selection Search Used in Wrapper

Best first forward selection search is an AI search strategy and more robust than hill-climbing. Best first forward selection search will select the most promising feature, which mostly improves the accuracy of the model from the features generated so far and not expanded. If the path being explored begins to appear less promising, it can back-track to a more promising previous subset and continue the search from there. To avoid searching the entire search space, the following stop criterion for best first forward selection search is used: if it can not find a feature that improve the estimation accuracy of the model in the last n expansions, it will stop and return the best solution so far (n in these experiments is set to 5 as it is the default value in WEKA [7]).

Ordinary Least Squares Regression

Ordinary Least Squares regression (LSR) method is the classical statistical approach of general linear regression modeling using least squares. It has been widely known and discussed extensively. LSR is the statistical procedure to estimate the linear relationship between the dependent variable Y (the prediction of the model) and the independent variables 1 *X* , .., *i X* , .. *n X* (the parameters of the model). LSR minimizes squared error with equation 23. *I* β is calculated with mathematical matrix algorithms with equation 24.

$$\boldsymbol{Y_i = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \cdots + \beta_k X_{ki} + \varepsilon_i}$$

NASA effort data used in this study are available on-line at the PROMOSE repository of public domain software engineering data set: http://promise.site.uottawa.ca/SERepository/datasets/cocomonasa.arff. For example, if LSR is applied LSR into COCOMO 81 linear model with the NASA 60 projects without eliminating the co-linear parameters and selecting attribute method, here is the linear regression model with 10-fold cross-validation:

ACT_EFFORT =
-1.1554 * AEXP +
2.7992 * DATA +
-0.1527 * PCAP +
-6.2363 * VEXP +
-1.5573 * MODP +
-1.1246 * RELY +
0.1986 * TOOL +
1.1126 * TURN +
0.8644 * CPLX +
2.4129 * LEXP +
1.3613 * SCED +
-0.6794 * VIRT +
3.2141 * ACAP +
77
-0.745 * STOR +
2.8783 * TIME +
1.0712 * LOC +
1.3462

Correlation coefficient is 0.9825. Mean absolute error is 0.1896. Root mean squared error is 0.2579. Relative absolute error is 16.0925 %. Root relative squared error is 18.3653 %. The accuracy for this linear regression model of COCOMO 81 are PRED(25)=70 and PRED(30)=75.

Summary

Presented mathematical model for the calibration of COCOMO model via reduction of it's main equation increases the accuracy of the model for specific company domain, but decreseas the accuracy of the model for the generalized case. The accuracy of calibration depends on the amount of historical data in the company that is used to calibrate the model. Some variation in calibrated model results may appear if the company changes the specifics of the software projects being developed. This may require recalibration of the model to accont for new specifics, or otherwise accuracy may be even worse than for the generalized COCOMO model.

References

1. R. Kohavi, G. John, "Feature Extraction, Construction and Selection : A Data Mining Perspective", edited by H. Liu and H. Motoda.

2. Foss, T.; Stensrud, E.; Kitchenham, B.; Myrtveit, I. "A Simulation Study of the Model Evaluation Criterion MMRE", IEEE Transactions on Software Engineering, 29(2003)11, pp. 985-995

3. M. Jørgensen, D. I. K. Sjøberg. "An effort prediction interval approach based on the empirical distribution of previous estimation accuracy",

Journal of Information and Software Technology, 45 (3), March 2003, pp. 123-136.

4. K. J. Molokken-Ostvold. "Effort and Schedule Estimation of Software Development Projects", PhD-thesis, 2004, http://www.simula.no/photo/effort_and_schedule_estimation_of_software_development_projects.pdf

5. S. Chulani, B. Boehm, B. Steece. "Bayesian analysis of empirical software engineering cost models". IEEE Transactions on Software Engineering, 25(4), July/August 1999.

6. T. Mitchell. "Machine Learning", McGraw Hill, ISBN 0070428077, 1997.

7. I. H. Witten, E. Frank, "Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations". Morgan Kaufmann, 1999.

8. M. Shepperd and C. Schofield., "Estimating Software Project Effort Using Analogies", IEEE Transactions on Software Engineering, Nov 1997, Vol. 23, No. 12. http://www.utdallas.edu/~rbanker/SE_XII.pdf

9. K. Srinivasan and D. Fisher. "Machine learning approaches to estimating software development effort", IEEE Trans. Soft. Eng., pages 126–137, February 1995.

10. G. Wittig, G. Finnie. "Estimating software development effort with connectionist models", Information and Software Technology, 39(7):469–476, 1997.

11. L. C. Briand, Kh. El Emam, D. Surmann, I. Wieczorek. "An assessment and comparison of common software cost estimation modeling techniques", The 21st International Conference on Software Engineering, May 1999.

12. C. Mair, G. Kadoda, M. Lefley. "An Investigation of Machine Learning Based Prediction Systems", Journal of software systems, vol. 53, pp. pp23-29, July 2000.

13. S. Bibi, I. Stamelos, L. Aggelis. "Bayesian Belief Networks as a Software Productivity Estimation Tool", 1st Balkan Conference in Informatics, Thessaloniki, Greece, November 2003.

14. G. Boetticher. "When will it be done? the 300 billion dollar question, machine learner answers", IEEE Intelligent Systems, June 2003.

15. T. Menzies, D. Port, Z. Chen, J. Hihn, S. Stukes. "Validation Methods for Calibrating Software Effort Models", ICSE 2005, May 15–21, 2005

16. M.A. Hall and G. Holmes. "Benchmarking attribute selection techniques for discrete class data mining". IEEE Transactions On Knowledge And Data Engineering, 15(6):1437– 1447, 2003.

Баценко Дмитро Володимирович - старший викладач кафедри інженерії програмного забезпечення факультету комп'ютерних наук Національного авіаційного університету.
Наукові інтереси: управління проектами, методи та моделі оцінки вартості розробки ПЗ, методології та технології розробки програмного забезпечення, тестування програмного забезпечення, якість та управління якістю ПЗ, бази даних, життєвий цикл ПЗ, програмне забезпечення роботів.
E-mail: dmytro.batsenko@gmail.com

Стаття надійшла до редакції 12.10. 2011 р.
Після доопрацювання – 16. 11. 2011 р.