Neural Comput & Applic (2011) 20:389–401
DOI 10.1007/s00521-010-0405-5

ORIGINAL ARTICLE

Applying a general regression neural network for predicting development effort of short-scale programs

Cuauhtemoc Lopez-Martin

Received: 1 November 2009 / Accepted: 4 June 2010 / Published online: 18 June 2010

Abstract Software development effort prediction is considered in several international software processes as the Capability Maturity Model-Integrated (CMMi), by ISO-15504 as well as by ISO/IEC 12207. In this paper, data of two kinds of lines of code gathered from programs developed with practices based on the Personal Software Process (PSP) were used as independent variables in three models for estimating and predicting the development effort. Samples of 163 and 80 programs were used for verifying and validating, respectively, the models. The prediction accuracy comparison among a multiple linear regression, a general regression neural network, and a fuzzy logic model was made using as criteria the magnitude of error relative to the estimate (MER) and mean square error (MSE). Results accepted the following hypothesis: effort prediction accuracy of a general regression neural network is statistically equal than those obtained by a fuzzy logic model as well as by a multiple linear regression, when new and change code and reused code obtained from short-scale programs developed with personal practices are used as independent variables.

Keywords Software engineering education and training · Software development effort prediction · General regression neural network · Fuzzy logic · Multiple linear regression · Personal software process

C. Lopez-Martin (✉)
Information Systems Department, CUCEA,
Guadalajara University, P.O. Box 45100,
Guadalajara, Jalisco, Mexico
e-mail: cuauhtemoc@cucea.udg.mx

1 Introduction

Planning, bidding, and budgeting of software development projects consider as an important factor the effort prediction required to complete the project; bad effort estimates may address to poor planning, low profitability, and consequently products with poor quality [23]. Software researchers have addressed the problems of effort estimation for software development projects since at least the 1960s, and the early models were those based upon statistical regression [24].

Software estimation has been identified as one of the three great challenges for half-century-old computer science [9], and several effort estimation techniques have been proposed and researched over the last years [7, 35]. Researchers aimed to (1) determine which technique has the greatest effort prediction accuracy or (2) propose new or combined techniques that could provide better estimates.

Software development estimation techniques can be classified into three general categories [35]:

1. Expert judgment: It remains the predominant methodology of choice [5]. The term *expert estimation* is not clearly defined and covers a wide range of estimation approaches; a common characteristic is, however, that intuitive processes constitute a determinant part of the estimation [21]. This technique implies a lack of analytical argumentation and by the frequent use of phrases such as "*I think that …*" and "*I feel that …*" [24], and it aims to derive estimates based on an experience of experts on similar projects. The means of deriving an estimate are not explicit and therefore not repeatable [35]. Psychological research on real-world quantitative expert estimation "has not culminated in any theory of estimation, not even in a

coherent framework for thinking about the process" [24].

2. Algorithmic models: It has been very popular in literature [8, 35]. It attempts to represent the relationship between effort and one or more characteristics of a project; the main cost driver in a model is usually taken to be some notion of software size (e.g., the number of lines of source code). Its general form is a linear regression equation, as that used by Kok [29], or a group of non-linear regression equations as those used by Boehm in COCOMO 81 [2] and COCOMO II [4].
3. Machine learning: Machine learning techniques have been used as a complement or alternative to the previous two techniques in recent years [40, 46]. Fuzzy logic models are included in this category [32] as well as neural networks [40], genetic algorithms [47], genetic programming [10], regression trees [24], case-based reasoning [25], and associative memories [30].

A primary conclusion of a previous research is that no single technique is best for all situations and that a careful comparison of the results of several approaches is most likely to produce realistic estimates [3]. Based upon this conclusion, in this paper, are compared the accuracies of a multiple linear regression (MLR), a general regression neural network (GRNN), and a fuzzy logic model (FLM). This comparison is based upon the two following main stages when an estimation model is used [37]: (1) the model adequacy checking or model verification (estimation stage) must be determined, that is, whether the model is adequate to describe the observed (actual) data; if so, then (2) the estimation model is validated using new data (prediction stage).

The hypothesis to be investigated in this paper is the following:

> Effort prediction accuracy of a general regression neural network is statistically equal or better than those obtained by a fuzzy logic model as well as by a multiple linear regression, when new and change code and reused code obtained from short-scale programs developed with personal practices are used as independent variables.

The foundation for predicting the effort should be based on the assumption of that unless software engineers have the capabilities provided by personal training, they cannot properly support their teams or consistently and reliably produce quality products; this assumption is from the Personal Software Process (PSP) whose practices and methods have been used by thousands of software engineers for delivering quality products on predictable schedule [43].

The capability maturity model (CMM) gives an available description of the goals, methods, and practices needed in software engineering industrial practice, while the PSP allows its instrumentation at a personal level (twelve of the eighteen key process areas of the CMM are at least partially considered in PSP) [17]. This paper is based upon some PSP practices and takes account of the guidelines suggested in [27].

In this study, two measures related to software size (lines of code) as well as to development time (effort) were gathered from 163 small programs developed by 53 programmers. From this set of programs, a MLR equation as well as a FLM was generated, and then, a GRNN was trained too; then, their adequacy was checked (verification). Afterward, these three models were validated with the effort estimation of 80 programs developed by other group integrated by thirty programmers. Verification data of this research were gathered through 2005, 2006, and 2007 years, whereas validation data were obtained through 2008 and first trimester of 2009 year.

1.1 Fuzzy logic

Newer computation techniques on cost estimation that are non-algorithmic appeared in the 1990s; fuzzy logic with its offerings of a powerful linguistic representation can represent imprecision in inputs and outputs, while providing a more expert knowledge-based approach to model building [1].

A fuzzy model is a modeling construct featuring two main properties [42]: (1) It operates at a level of linguistic terms (fuzzy sets) and (2) it represents and processes uncertainty.

Estimation techniques have an important limitation, which arises when software projects are described using categorical data (nominal or ordinal scale) such as *small*, *medium*, *average*, or *high* (linguistic terms or values). A more comprehensive approach to deal with linguistic values is using the fuzzy set theory [20, 34]. Specifically, fuzzy logic offers a particularly convenient way to generate a keen mapping between input and output spaces thanks to the natural expression of fuzzy rules [49].

In software development effort estimation, two considerations justify the decision of implementing a fuzzy model: first, it is impossible to develop a precise mathematical model of the domain [31]; second, metrics only produce estimations of the real complexity. Thus, according to the previous assertions, formulating a tiny set of natural rules describing underlying interactions between the software metrics and the effort estimation could effortlessly reveal their intrinsic and wider correlations. Disadvantages of a fuzzy model could be that (1) it requires a lot of data, (2) the estimators must be familiar with the historically developed programs, and (3) it is not useful for programs much larger or smaller than the historical data [17]. In this research, only the third one represents a

disadvantage since the sample was conformed by 163 programs and those were well known to the estimator.

In this paper, data defuzzification is constructed based on a rule induction system replacing the crisp facts with fuzzy inputs, then an inference engine uses a base of rules to map inputs to a fuzzy output which either can be translated back to a crisp value or left as a fuzzy value [44].

1.2 General regression neural network

There are some advantages when using estimation by artificial neural networks [40] which are as follows: (1) it allows the learning from previous outcomes and (2) it can model a complex set of relationships between the dependent variable (effort) and the independent variables (i.e., new and changed, as well as reused lines of code); moreover, they have been considered as promising techniques to build predictive models, because they are capable of modeling non-linear relationships [26]; however, there are some shortcomings which include the following: (1) the ability of neural networks to solve problems of high complexity has been proven in classification and categorization areas, whereas in the cost estimation field, we deal with a generalization rather than a classification problem and (2) there is not guidelines for the construction of the neural network topologies (number of layers, number of neurons by layer, or initial weights).

This paper uses a general regression neural network (GRNN) whose principal advantages are (a) fast learning and (b) convergence to the optimal regression surface as the number of samples becomes very large. The GRNN has shown that even with sparse data in a multidimensional measurement space, the algorithm provides smooth transitions from one observed value to another [45].

Figure 1 shows the architecture of a GRNN [45]. The input units are merely distribution units that provide for all the scaled measurement variables X to all neurons on the second layer, the pattern units that are dedicated to one exemplar or one cluster center. When a new vector X is entered into the network, it is subtracted from the stored vector representing each cluster center. Either the squares or the absolute values of the differences are summed and fed into a non-linear activation function. The activation function normally used is the exponential. The pattern units' output is passed onto the summation units. The summation units perform a dot product between a weight vector and a vector composed of the signals from the pattern units. The summation unit that generates an estimate of $F(X)K$ sums the outputs of the pattern units weighted by the number of observations each cluster center represents. The summation unit that estimates $Y'\ F(X)K$ multiplies each value from a pattern unit by the sum of the samples Y^j associated with cluster center X^i. The output

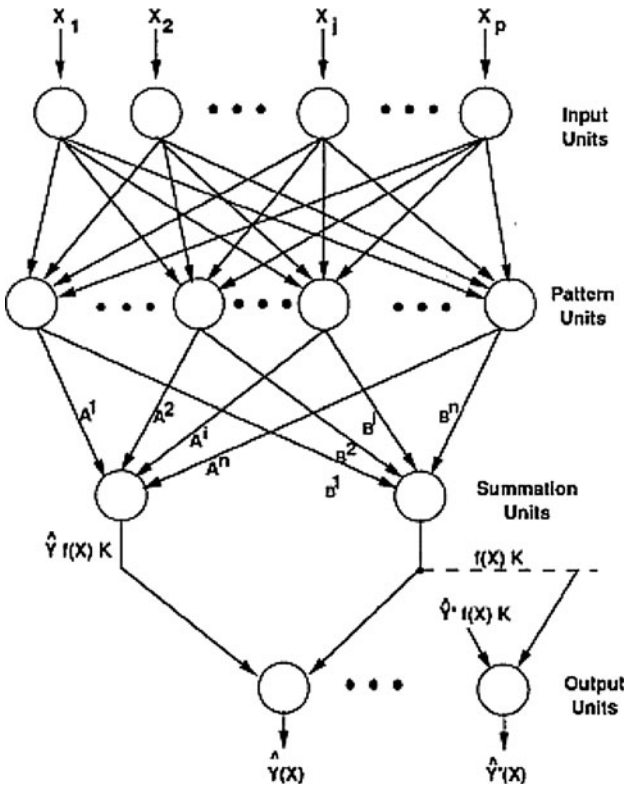

Fig. 1 General regression neural network diagram

unit merely divides $Y'\ F(X)K$ by $F(X)K$ to yield the desired estimate of Y. When estimation of a vector Y is desired, each component is estimated using one extra summation unit, which uses as its multipliers sums of samples of the component of Y associated with each cluster center X^i. Figure 1 depicts a feedforward network that can be used to estimate a vector Y from a measurement vector X.

1.3 Software measurement (independent variables)

In spite of the availability of a wide range of software product size measures, source lines of code (LOC) remains in favor of many models [33, 35]. In fact, since attributes must be relevant for the effort estimation, researches use to correlate lines of code to effort [19].

There are two measures of source code size: physical source lines and logical source statements. The count of physical lines gives the size in terms of the physical length of the code as it appears when printed [39].

In this study, the independent variables of the models are new and changed (N&C) as well as reused code, and all of them are considered as physical lines of code (LOC). N&C is composed of added and modified code. The added code is the LOC written during the current programming process, while the modified code is the LOC changed in the base program when modifying a previously developed

program. The base program is the total LOC of the previous program, while the reused code is the LOC of previously developed programs that are used without any modification [17].

A coding standard should establish a consistent set of coding practices that is used as a criterion when judging the quality of the produced code [17]. Hence, it is necessary to always use the same coding and counting standards. The programs developed of this study followed these guidelines.

1.4 Accuracy criteria

It has been demonstrated that the magnitude of relative error, or MRE, (a common criterion for the evaluation of cost estimation models) does not identify the best prediction model [14]. In accordance with [14], the implications of this finding are that the results and conclusions on prediction models over the past 15–25 years are unreliable and may have misled the entire software engineering discipline; therefore, they strongly recommend not using MRE to evaluate and compare prediction models, but the magnitude of error relative to the estimate, or MER, that was proposed in [28].

The MER is defined as follows:

$$\mathrm{MER}_i = \frac{|\mathrm{Actual\ Effort}_i - \mathrm{Estimated\ Effort}_i|}{\mathrm{Estimated\ Effort}_i}$$

The MER value is calculated for each observation i whose effort is estimated. The aggregation of MER over multiple observations (n) can be achieved through the mean (MMER) as follows:

$$\mathrm{MMER} = (1/n)\sum_{i=1}^{n}\mathrm{MER}_i$$

Intuitively, MER seems preferable to MRE since MER measures the error relative to the estimate. Results of MMER in [14] had better results than MMRE; this fact is the reason for using MMER in this study.

The accuracy of an estimation technique is inversely proportional to the MMER. In several papers, a MMRE ≤ 0.25 has been considered as acceptable, however, who has proposed this value [11] neither present any reference to studies nor any argumentation providing evidence [22]. On the other hand, a reference for an acceptable value of MMER has not been found.

Another complementary criterion used in this study for evaluating the accuracy of the models is the mean square error (MSE) which is calculated for n-programs as follows:

$$\mathrm{MSE}_i = \frac{(\mathrm{Actual\ Effort}_i - \mathrm{Estimated\ Effort}_i)^2}{n}$$

1.5 Related work

Artificial neural networks and statistical regression have been investigated, and results have shown that the performance of both techniques indicates that they are competitive with models generated from data of large programs [12, 13, 15]. The feedforward multi-layer perceptron with back propagation learning algorithm is the most commonly used in the effort estimation field [40]. However, it has not found any paper for which a GRNN has been used for predicting the software development effort of short-scale programs. One paper found uses a GRNN but for predicting the number of defects in a class as well as for predicting the number of lines changed by class [36].

On the other hand, fuzzy logic has been used for predicting the development effort of large programs [1, 6, 16, 19, 39, 48], whereas about short-scale programs only one paper was found; however, it did not use new and changed (N&C) as well as reused code as independent variables, but only one of them (N&C) [32].

1.6 Verification (estimating) and validation (predicting) of models

There are two main stages for using an estimation model [37]:

1. Model adequacy checking (or model verification): The MMER is calculated for each model as well as an analysis of variance (ANOVA) to compare their MER.
2. Model validation: Once the adequacy of the models is checked, the effort of the new gathered dataset is predicted.

This research involved a sample integrated by 83 programmers divided into two groups: one for checking the adequacy of the models (53 developers with 163 programs) and the other (30 developers with 80 programs) for validating the models.

2 Experimental design

The experiment was done inside a controlled environment having the following characteristics:

- Each programmer developed seven small programs. Seven was a number established because of the availability of developers (the course duration is a principal concern for industrial organizations [18]).
- Ten sessions were carried out by programmer. The first session was spent for the introduction to PSP and for

studying and making the coding and counting standards. From the second to the eighth training day session, one program was developed (one daily). Finally, the ninth and tenth days were assigned to make final reports.

- In all programs, the following artefacts were used by all developers [17]: coding standard, counting standard (see Table 1), defect-type standard, project plan summary, time recording log, defect recording log, and process improvement proposal. A test report template was introduced in the second through the seventh programs in the testing phase. Code review checklist was introduced in the third–seventh programs, and design review checklist was used in the fourth–seventh programs. Thus, from fourth program on, all practices and logs planned for this study were used by all the developers. Hence, the first, second, and third programs were excluded from this study, otherwise the comparison of the development time results would have been unfair.
- Developers had already received at least a course about the imperative programming language that they used though the assignments–programs.
- Developers were constantly supervised and advising about the process.
- Each developer selected his/her own imperative programming language (as C and Pascal) whose code standard had the following characteristics: each compiler directive, variable declaration, constant definition, delimiter, assign sentence, as well as flow control statement, was written in a line of code.
- The code wrote in each program was designed by the developers to be reused in next programs.
- All programs were developed based upon the following phases of the process: planning, algorithm design, coding, compiling, testing, and postmortem, and from the fourth program, two phases were added. The development time considered for this study included only the following phases: algorithm design, design review, coding, code review, compiling, and testing.
- The kind of the developed programs had a similar complexity of those suggested in [17]. From a set of 18 small programs, a subset of seven was randomly assigned to each of the all programmers. A brief description of these 18 programs is the following:

1. Estimating the mean of a sample of n real numbers.
2. Estimating the standard deviation of a sample of n real numbers.
3. Matrix addition integrated by real numbers.
4. Summing the diagonal of a real numbers square matrix.
5. Translating from a quantity to letters.
6. Calculating the correlation (r) between two series of real numbers.
7. Computing the linear regression equation parameters a and b ($y = a+bX$).
8. Calculating z-values from a sample of real numbers.
9. Calculating the size of a sample.
10. Calculating the y-values from a sample of real numbers using the normal distribution equation.
11. Calculating the estimation standard error (from $y = a+bX$).
12. Calculating the coefficient of determination (r^2) from a linear regression equation.
13. Calculating both upper and lower limits from a sample of real numbers based upon its standard deviation and average.
14. Calculating the coefficient of variation from a distribution.
15. Estimating the values based upon statistical empirical rule.
16. Counting the physical lines of code of a program omitting comments and blank lines.
17. Both storing and searching records from a file.
18. Both deleting and modifying records from a file.

Table 1 Counting standard

Count type	*Type*
Physical/logical	Physical
Statement type	*Included*
Executable	Yes
Non-executable	
Declarations and	Yes (one by text line)
Compiler directives	Yes (one by text line)
Comments and Blank lines	No
Delimiters	
*{*and*}* or *begin* and *end*	Yes

3 Conducting the experiment

The number of programs developed by the 53 programmers was 371. One hundred and fifty-nine of the 371 were excluded because they corresponded to the first, second, or third program. From those 212 programs, in 175 of them, lines of code were reused. In addition, 12 programs of those 175 were excluded because they presented errors in their time recording. Hence, 163 was the final number that represents the sample for verifying the three models. In Appendices 1 and 3, actual data and names by developer are depicted, respectively.

Table 2 Multiple linear regression analysis

Source	Sum of squares	Degrees of freedom	Mean square	*F* ratio	*p* Value	Parameter	Estimated	*t* Statistic	*p* Value
(a) ANOVA						(b) Parameters			
Model	132376.0	2	66187.8	106.73	0.000	Constant	44.7136	11.0923	0.0000
Residual	99220.7	160	620.129			N&C	1.08075	14.5973	0.0000
Total	231596.0	162				Reused	−0.14543	−2.28037	0.0239

3.1 Regression model

The following multiple linear regression equation considering new and changed (N&C) as well as reused code was generated from Appendix 1:

$$\text{Effort} = 44.7136 + (1.08075 \times \text{N\&C}) - (0.1454 \times \text{Reused})$$

In accordance with [17], a $r^2 \geq 0.5$ (coefficient of determination) is an acceptable value for predicting. In this case, the r^2 of this equation was 0.57. ANOVA for this equation (Table 2a) shows a statistically significant relationship between the variables at the 99% confidence level.

However, an ANOVA was not sufficient. To determine whether the model could be simplified a parameters analysis of the multiple linear regression was done. Table 2b depicts the results for this analysis, and the highest *p*-value on the independent variables is 0.0239, belonging to reused code. Since this *p*-value is less than 0.05, reused code is statistically significant at the 95% confidence level. Consequently, the independent variable of reused code was not removed. Hence, this variable will have to be considered for its evaluation in the GRNN as well as in the fuzzy logic model.

3.2 Fuzzy rules

The term "fuzzy identification" usually refers to the techniques and algorithms for constructing fuzzy models from data. There are two main approaches for obtaining a fuzzy model from data [48]:

1. The expert knowledge is translated in a verbal form into a set of if–then rules. A certain model structure can be created, and parameters of this structure, such as membership functions and weights of rules, can be tuned using input and output data.
2. No prior knowledge about the system under study is initially used to formulate the rules, and a fuzzy model is constructed from data based on a certain algorithm. It is expected that extracted rules and membership functions can explain the system behavior. An expert can modify the rules or supply new ones based upon his or her own experience. The expert tuning is optional in this approach.

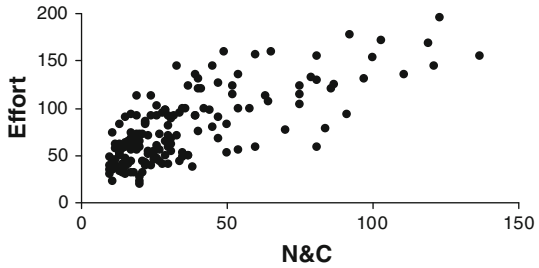

Fig. 2 Effort-N&C scatter plot ($r = 0.74$)

This paper is based upon the first approach. The fuzzy rules were formulated as follows:

1. Correlation analysis between N&C and effort. Figure 2 shows that the higher the value of N&C, the higher is the effort. Correlation value (r) is 0.74. From this correlation pattern, the following rules can be formulated.

   Rule 1: If (*New&Changed* is *Small*), then *Effort* is *Low*
   Rule 2: If (*New&Changed* is Big), then *Effort* is *High*

2. Correlation between reused code and effort from Appendix 1 having the table ordered by N&C. Data of N&C, reused code, and effort from Appendix 1 were put in ascending order (having N&C as first criterion for ordering it). Then, the data of this table were divided in two subsamples *A* and *B*. *A* incorporated by the N&C smaller programs, and *B* incorporated by the bigger ones. Then, reused code and effort from subsamples *A* and *B* (first 81 programs and last 82 programs, respectively) were correlated with effort. Figure 3a, b depicts these two scatter plots. Figure 3a shows that if reused code is small or big, the effort tends to be low in relation to programs of Fig. 3b, while in Fig. 3b it can be seen that if reused code is small or big, the effort tends to be higher in relation to programs of Fig. 3a.

Hence, rules 1 and 2 can be completed as follows:

Rule 1: If (*New&Changed* is *Small*) and (*Reused* is *Small or Reused* is *Big*), then *Effort* is *Low*
Rule 2: If (*New&Changed* is Big) and (*Reused* is *Small or Reused* is *Big*), then *Effort* is *High*

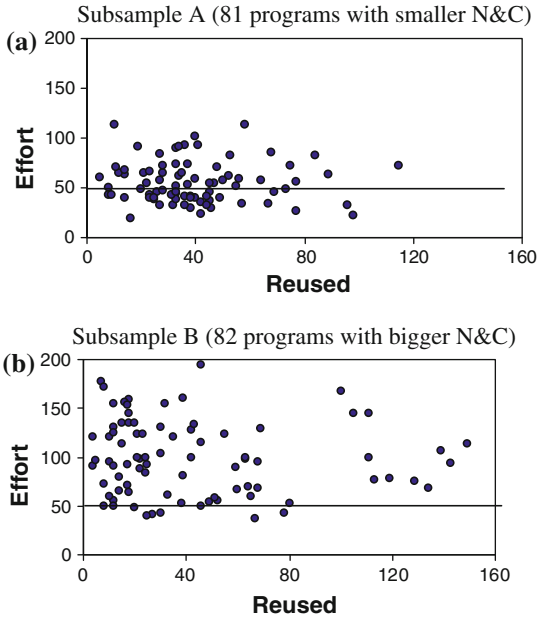

Fig. 3 Effort-Reused code scatter plots

Implementing a fuzzy system requires that the different categories of the different inputs be represented by fuzzy sets, which in turn is represented by membership functions (MF) [1]. The MF type considered in this experiment was triangular because it has demonstrated better accuracy than others like Gaussian and trapezoidal types [48].

A triangular MF is a three-point (parameters) function, defined by minimum (a), maximum (c), and modal (m) values, that is, MF(a, m, c) where $a \leq m \leq c$. Their scalar parameters (a, m, c) are defined as follows:

$\text{MF}(x) = 0$	$\text{MF}(x) = 1$	$\text{MF}(x) = 0$
if $x < a$	if $x = m$	if $x > c$

The values of MF parameters by fuzzy model were then defined. From values close or equal to both minimum and maximum of both program code sizes and efforts, the parameters were iteratively adjusted until obtaining the best (smallest) MMER possible. Table 3 shows the final values of the model parameters.

4 Analysis

4.1 Model adequacy checking (model verification)

Multiple linear regression equation (depicted in Sect. 3.1), the fuzzy logic model (depicted in Sect. 3.2), and regression neural network (depicted in Sect. 1.2) were applied to original data set, and the MER by program as well as the MMER by technique (model) was then calculated (see Appendix 1). The MMER results by technique were the following: multiple linear regression = 0.27, general regression neural network = 0.24, and fuzzy logic model = 0.25.

Table 3 Parameters of the fuzzy model membership functions

Type of variable	Variable	MF	Parameters		
			A	*m*	*b*
Input	N&C	Small	1	1	51
		Big	1	150	150
	Reused	Small	1	9	132
		Big	17	50	150
Output	Effort	Low	1	1	51
		High	26	191	200

Table 4 MER ANOVA (verification of models)

Source	Sum of squares	Degrees of freedom	Mean square	*F* ratio	*p* Value
Between groups	0.08808	2	0.04404	1.29	0.2760
Within groups	16.5851	486	0.03412		
Total	16.6732	488			

The ANOVA for MER of the programs (Table 4) shows that there is not a statistically significant difference among the accuracy of prediction for the three techniques at the 95.0% confidence level. This result can graphically be interpreted based upon the means plot of Fig. 4b. In addition, in accordance with Shapiro–Wilks test, a normal probability plot of the residuals should be roughly linear as it is shown in Fig. 4a.

In accordance with MSE criterion, ANOVA for models showed a p-value of 0.005, that is, there was a statistically significant difference among the MSE of models at the 95.0% confidence level. Means plot of Fig. 5 shows that the GRNN (having the lower MMER) has difference with FLM, whereas the GRNN does not have difference with MLR.

4.2 Model validation

Other group integrated by thirty programmers developed 210 programs. Ninety of the 210 were not considered because they corresponded to the first, second, or third program. Eighty of those 120 programs correspond to the sample size for validating the models (once that only programs reusing code were selected, and once programs having errors in their time recording were excluded). In Appendix 2, actual data by developer are depicted.

The three models for estimating the effort were applied using the data depicted in Appendix 2, the MMER results by

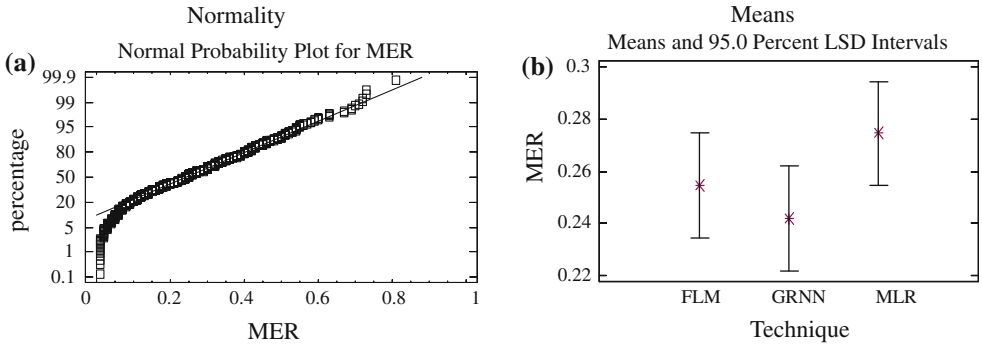

Fig. 4 Plots of MER ANOVA (verification stage)

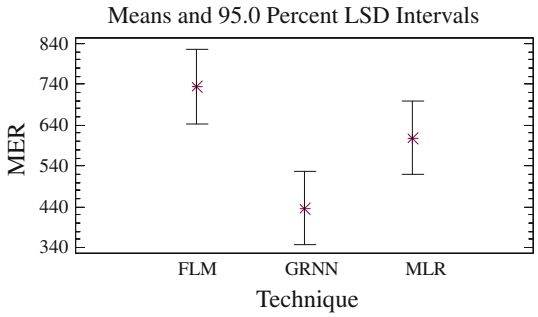

Fig. 5 Plot of MSE ANOVA (verification stage)

technique were the following: multiple linear regression = 0.29, general regression neural network = 0.31, and fuzzy logic model = 0.31. In accordance with the ANOVA for MMER models (Table 5), there is not a statistically significant difference among the accuracy of prediction for the three models at the 95.0% confidence level (Fig. 6b). Figure 6a shows that residuals related to normality data of this ANOVA are met (Shapiro–Wilks test).

MSE ANOVA for models showed a p-value of 0.8627, that is, there was not a statistically significant difference among the MSE of FLN, GRNN, and MLR models at the 95.0% confidence level.

5 Discussion

This study is related to software engineering education and training (SEET). Discipline about the process for software organizations is one of the SEET goals. In accordance with [43], the performance of a development organization is determined by the performance of its engineering teams; and the performance of an engineering team is determined by the performance of the team members; and the performance of the engineers is, at least in part, determined by the practices these engineers follow in doing their work. This assertion is also supported by [17], in the sense of that unless software developers have the capabilities provided by personal training, they cannot properly support their teams or consistently and reliably produce quality products. One of those practices is the software development effort prediction that can be started at individual level developing components that could be integrated when large systems are built. Effort prediction is one of the three main practices used for training developers at personal level (the other two are related to software defects and software size [43]). Prediction techniques at personal level that have been applied are related to expert judgment, statistical regression, and fuzzy logic, and the discipline of using these techniques has even been scaled to developing large systems. Because no single technique is best for all situations and a careful comparison of the results of several approaches is most likely to produce realistic estimates [3], this paper applied an additional one: a GRNN. Neural networks have already been applied for predicting effort too [12, 13, 15, 40]; however, neither of them has been a GRNN. Results of this paper could encourage researchers to analyze the prediction accuracy when a GRNN is applied for predicting effort not only of developed software components at individual level, but of large systems built by teams of developers. In addition, models of this research used two measures of software size, whereas future research could involve the GRNN application involving more independent variables related to large systems (as those considered by statistical regression models [4]).

Table 5 MER ANOVA (validation of models)

Source	Sum of squares	Degrees of freedom	Mean square	F ratio	p Value
Between groups	0.02177	2	0.01088	0.29	0.7460
Within groups	8.79456	237	0.03710		
Total	8.8163	239			

6 Conclusions and future research

Levels of software engineering education and training could be classified in the small as well as in the large, this paper focused its interests on individual training based on short-scale programs and using PSP whose practices and methods

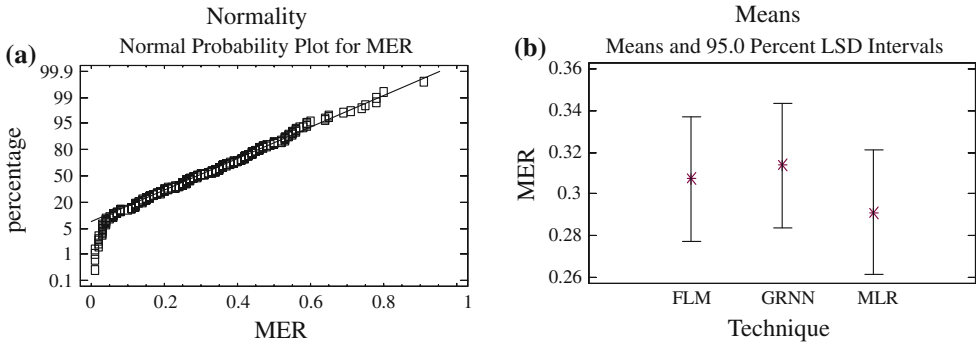

Fig. 6 Plots of MER ANOVA (validation stage)

are used for delivering quality products on predictable schedules. This paper approached its focus on the following practice: estimation and prediction of the software development effort. An accuracy comparison among a multiple linear regression (MLR), a general regression neural network (GRNN), and a fuzzy logic model (FLM) was depicted. In this research, a total of 163 programs were developed by a group of 53 programmers. Using the data gathered (two types of lines of code as well as effort) from these programs, three models were applied for estimating the effort. Then, models were validated with data gathered of eighty programs developed by other group of thirty developers. All programs were developed with personal practices from PSP. In verification and validation stages, results from the two machine learning techniques were compared with those of a multiple linear regression. This comparison was based on MER as well as on MSE. In validation stage, the GRNN had a same statistically significant difference than FLM and MLR. These results allow accepting the following hypothesis: in linear case, neural network is statistically equal to those obtained by a fuzzy logic model as well as by multiple linear regression, which suggests that a general regression neural network could be used for predicting the development effort when new and change code and reused code obtained from short-scale programs developed with personal practices are used as independent variables.

Future research involves the predictive accuracy comparison applying a GRNN for data from large systems built by teams of developers. Moreover, using classifiers and associative memories based on datasets from individual as well as from team projects.

Acknowledgments Author of this paper would like to thank CUCEA of Guadalajara University, Jalisco, México, Programa de Mejoramiento del Profesorado (PROMEP), as well as to Consejo Nacional de Ciencia y Tecnología (Conacyt).

Appendix 1

Actual data by developer from design to testing phases for generating and verifying the models: *P*: Number of program, DP: Developer (see Appendix 3), N&C: New and Changed code, R: Reused code, AE: Actual Effort (minutes), MER: Magnitude of Error Relative to the estimate, MLR: Multiple Linear Regression, GRNN: General regression neural network; FLM: Fuzzy Logic Model (Table 6).

Table 6

P	DP	N&C	R	AE	MER			*P*	DP	N&C	R	AE	MER			*P*	DP	N&C	R	AE	MER		
					MLR	GRNN	FLM						MLR	GRNN	FLM						MLR	GRNN	FLM
1	C1	22	84	82	0.46	0.41	0.25	56	P10	60	51	58	0.43	0.35	0.06	111	P28	19	28	72	0.18	0.27	0.19
2	C2	17	77	56	0.08	0.03	0.19	57	P10	45	18	145	0.60	0.40	0.74	112	P28	17	14	68	0.11	0.15	0.23
3	C2	17	89	63	0.26	0.19	0.35	58	P10	81	65	59	0.52	0.37	0.51	113	P28	15	26	46	0.20	0.17	0.34
4	C3	33	64	70	0.02	0.06	0.08	59	P11	50	24	83	0.13	0.26	0.02	114	P29	15	14	63	0.07	0.09	0.18
5	C3	58	21	99	0.05	0.18	0.05	60	P11	91	143	93	0.24	0.00	0.30	115	P29	24	10	113	0.63	0.73	0.27
6	C3	35	63	100	0.36	0.34	0.32	61	P12	36	24	100	0.25	0.27	0.28	116	P29	26	5	60	0.17	0.11	0.31
7	C4	75	23	124	0.01	0.02	0.10	62	P12	17	21	65	0.08	0.13	0.39	117	P29	16	14	40	0.33	0.32	0.59
8	C4	32	17	92	0.20	0.22	0.16	63	P12	42	111	100	0.35	0.08	0.16	118	P30	27	30	43	0.38	0.31	0.27
9	C4	20	40	59	0.02	0.06	0.01	64	P13	75	30	104	0.14	0.19	0.08	119	P30	25	33	45	0.33	0.25	0.15
10	C5	15	50	58	0.08	0.07	0.24	65	P14	81	32	155	0.21	0.18	0.28	120	P30	30	18	64	0.14	0.11	0.56
11	C6	13	32	33	0.39	0.38	0.29	66	P14	119	100	168	0.06	0.04	0.16	121	P30	11	45	45	0.10	0.11	0.35
12	C6	30	25	92	0.25	0.36	0.20	67	P15	103	8	171	0.10	0.07	0.22	122	P31	40	10	120	0.39	0.30	0.01

Table 6 continued

P	DP	N&C	R	AE	MER			*P*	DP	N&C	R	AE	MER			*P*	DP	N&C	R	AE	MER		
					MLR	GRNN	FLM						MLR	GRNN	FLM						MLR	GRNN	FLM
13	C7	10	73	48	0.07	0.11	0.05	68	P15	97	30	131	0.10	0.05	0.04	123	P31	39	15	135	0.59	0.50	0.35
14	C7	12	69	45	0.06	0.20	0.02	69	P16	37	21	123	0.51	0.47	0.55	124	P31	40	12	131	0.52	0.42	0.19
15	C8	20	75	72	0.30	0.18	0.24	70	P16	16	12	64	0.06	0.09	0.37	125	P31	15	33	90	0.60	0.67	0.42
16	C8	21	115	72	0.42	0.04	0.03	71	P16	49	18	159	0.67	0.42	0.86	126	P32	19	58	113	0.99	0.81	0.45
17	C9	34	10	95	0.19	0.19	0.18	72	P16	123	46	195	0.14	0.00	0.34	127	P32	31	22	88	0.17	0.24	0.60
18	C9	137	12	155	0.19	0.00	0.07	73	P17	41	35	121	0.44	0.48	0.63	128	P32	13	53	83	0.63	0.53	0.15
19	C10	22	68	85	0.45	0.27	0.30	74	P17	81	69	129	0.06	0.34	0.07	129	U1	92	7	177	0.24	0.19	0.38
20	C10	111	20	135	0.17	0.06	0.06	75	P18	29	63	98	0.47	0.32	0.31	130	U1	27	8	72	0.01	0.05	0.18
21	C10	87	12	125	0.09	0.11	0.02	76	P18	63	150	113	0.24	0.02	0.09	131	U2	13	20	49	0.12	0.12	0.02
22	C10	28	68	95	0.46	0.32	0.24	77	P19	75	46	115	0.03	0.06	0.04	132	U2	86	4	121	0.12	0.14	1.13
23	P1	17	28	65	0.10	0.17	0.39	78	P19	44	5	97	0.06	0.01	0.15	133	U3	35	20	48	0.40	0.40	0.46
24	P1	22	27	84	0.30	0.43	0.28	79	P19	100	17	153	0.02	0.02	0.11	134	U3	30	25	40	0.46	0.41	0.66
25	P1	30	17	71	0.05	0.01	0.09	80	P19	121	111	144	0.10	0.05	0.01	135	U3	28	27	41	0.42	0.36	0.13
26	P1	13	37	53	0.01	0.02	0.14	81	P20	79	43	133	0.07	0.04	0.06	136	U3	30	10	59	0.22	0.19	0.60
27	P2	47	42	127	0.42	0.37	0.80	82	P20	84	119	78	0.34	0.00	0.37	137	U4	20	77	27	0.51	0.55	0.79
28	P2	47	59	90	0.04	0.12	0.21	83	P21	18	44	32	0.45	0.41	0.32	138	U4	38	67	37	0.51	0.49	0.93
29	P2	17	96	32	0.35	0.30	0.31	84	P21	15	38	30	0.46	0.43	0.36	139	U4	20	98	22	0.58	0.55	0.33
30	P3	34	78	43	0.39	0.32	0.47	85	P21	20	16	19	0.70	0.69	0.67	140	U5	64	139	107	0.14	0.01	0.05
31	P3	35	38	52	0.32	0.25	0.26	86	P21	37	12	50	0.40	0.42	0.38	141	U5	30	134	68	0.18	0.04	0.06
32	P3	54	52	56	0.41	0.35	0.13	87	P22	20	35	65	0.06	0.16	0.12	142	U5	40	129	75	0.08	0.01	0.05
33	P3	50	80	52	0.40	0.05	0.41	88	P22	29	8	49	0.35	0.32	0.37	143	U5	70	113	77	0.26	0.01	0.39
34	P4	11	33	44	0.15	0.16	0.04	89	P22	22	9	42	0.37	0.34	0.36	144	U6	22	25	40	0.38	0.33	0.48
35	P4	25	8	50	0.29	0.25	0.31	90	P22	12	27	57	0.06	0.06	0.24	145	U6	25	45	54	0.17	0.09	0.46
36	P4	14	27	32	0.43	0.41	0.31	91	P23	24	38	39	0.40	0.33	0.43	146	U6	31	68	68	0.00	0.07	0.25
37	P5	31	33	61	0.17	0.06	0.16	92	P23	15	44	41	0.25	0.22	0.12	147	U7	26	28	47	0.32	0.24	0.54
38	P5	12	25	38	0.30	0.30	0.18	93	P23	22	23	39	0.40	0.35	0.40	148	U7	15	38	41	0.26	0.23	0.54
39	P5	23	22	54	0.19	0.12	0.21	94	P23	21	36	32	0.49	0.43	0.49	149	U7	17	23	43	0.28	0.24	0.63
40	P5	10	42	35	0.29	0.30	0.23	95	P24	19	47	55	0.06	0.01	0.03	150	U7	14	36	41	0.25	0.22	0.01
41	P6	19	56	59	0.03	0.03	0.11	96	P24	31	49	54	0.24	0.17	0.02	151	U8	52	15	114	0.15	0.02	0.31
42	P6	26	40	102	0.52	0.71	0.52	97	P24	27	12	55	0.24	0.20	0.27	152	U8	19	19	91	0.46	0.54	0.27
43	P6	23	11	71	0.04	0.11	0.04	98	P24	21	31	43	0.32	0.25	0.31	153	U8	39	4	91	0.05	0.03	0.29
44	P6	11	33	74	0.43	0.42	0.61	99	P25	37	46	50	0.36	0.29	0.18	154	U8	39	12	91	0.07	0.01	0.39
45	P7	11	42	23	0.54	0.54	0.50	100	P25	54	42	100	0.03	0.00	0.30	155	U9	47	60	67	0.23	0.16	0.12
46	P7	15	8	42	0.30	0.29	0.10	101	P25	15	33	38	0.32	0.29	0.06	156	U9	52	55	124	0.33	0.46	0.14
47	P7	23	33	51	0.21	0.12	0.25	102	P26	14	64	57	0.13	0.04	0.74	157	U10	11	45	37	0.26	0.27	0.37
48	P8	26	34	91	0.34	0.51	0.27	103	P26	19	48	71	0.22	0.27	0.51	158	U10	29	14	65	0.12	0.09	0.12
49	P8	17	41	93	0.63	0.73	0.98	104	P26	30	39	81	0.13	0.29	0.02	159	U10	13	55	51	0.00	0.08	0.08
50	P8	19	23	66	0.07	0.14	0.24	105	P26	14	67	34	0.32	0.42	0.30	160	U11	33	105	145	1.23	0.36	0.12
51	P8	13	34	62	0.15	0.17	0.34	106	P27	24	36	92	0.41	0.57	0.28	161	U11	65	39	160	0.46	0.32	0.29
52	P9	11	57	34	0.30	0.38	0.26	107	P27	60	16	156	0.45	0.25	0.39	162	U11	45	14	80	0.12	0.23	0.71
53	P9	12	52	62	0.24	0.17	0.34	108	P27	29	22	98	0.35	0.44	0.16	163	U11	54	18	135	0.34	0.14	0.11
54	P9	12	40	39	0.25	0.24	0.15	109	P27	10	49	39	0.19	0.23	0.08								
55	P9	10	46	30	0.39	0.40	0.34	110	P28	17	37	73	0.27	0.35	0.28			MMER			0.27	0.24	0.25

Appendix 2

Actual data by developer from design to testing phases for validating the models: *P*: Number of program, DP: Developer (Appendix 3), N&C: New and Changed code, R: Reused code, AE: Actual Effort (minutes), MER: Magnitude of Error Relative to the estimate, MLR: Multiple Linear Regression, GRNN: General regression neural network; FLM: Fuzzy Logic Model (Table 7).

Table 7

P	DP	N&C	R	AE	MER			*P*	DP	N&C	R	AE	MER			*P*	DP	N&C	R	AE	MER		
					MLR	GRNN	FLM						MLR	GRNN	FLM						MLR	GRNN	FLM
1	M1	23	38	54	0.16	0.06	0.26	28	M12	43	17	89	0.00	0.10	0.19	55	M21	70	32	70	0.40	0.45	0.46
2	M1	41	12	71	0.19	0.25	0.33	29	M12	25	30	71	0.05	0.18	0.08	56	UM1	80	47	168	0.35	0.36	0.28
3	M2	76	1	74	0.42	0.44	0.26	30	M12	48	13	58	0.39	0.47	0.52	57	UM1	39	37	117	0.44	0.53	0.14
4	M3	19	70	35	0.36	0.44	0.47	31	M13	40	29	79	0.06	0.06	0.24	58	UM1	97	12	33	0.78	0.78	0.75
5	M3	66	21	84	0.26	0.33	0.35	32	M13	30	22	84	0.14	0.21	0.02	59	UM2	117	63	202	0.25	0.04	0.48
6	M4	23	52	50	0.19	0.18	0.32	33	M13	19	40	68	0.14	0.24	0.03	60	UM2	96	35	107	0.25	0.21	0.20
7	M4	35	27	69	0.12	0.07	0.27	34	M13	64	8	53	0.53	0.59	0.59	61	UM3	99	54	83	0.42	0.26	0.38
8	M4	16	51	84	0.54	0.52	0.41	35	M14	71	37	74	0.36	0.42	0.43	62	UM3	94	36	63	0.55	0.53	0.53
9	M5	22	57	61	0.01	0.06	0.15	36	M14	31	40	122	0.69	0.91	0.39	63	UM3	143	15	40	0.80	0.74	0.71
10	M5	13	52	67	0.31	0.25	0.26	37	M14	71	16	77	0.35	0.40	0.41	64	UM4	98	70	100	0.29	0.03	0.26
11	M5	40	46	61	0.25	0.20	0.42	38	M15	104	29	85	0.44	0.39	0.37	65	UM4	32	54	59	0.17	0.15	0.34
12	M6	12	22	53	0.03	0.03	0.04	39	M15	36	59	85	0.13	0.13	0.12	66	UM5	81	68	115	0.06	0.21	0.12
13	M6	23	9	38	0.44	0.41	0.48	40	M16	55	34	116	0.17	0.04	0.08	67	UM5	70	27	97	0.17	0.23	0.25
14	M6	25	10	45	0.36	0.32	0.42	41	M16	86	18	87	0.36	0.36	0.34	68	UM6	70	52	130	0.15	0.26	0.01
15	M7	41	21	111	0.29	0.19	0.04	42	M17	64	23	90	0.19	0.27	0.30	69	UM6	52	24	112	0.15	0.02	0.11
16	M8	30	31	106	0.46	0.64	0.23	43	M18	78	52	115	0.05	0.02	0.12	70	UM6	77	31	57	0.54	0.56	0.56
17	M8	65	6	132	0.16	0.02	0.03	44	M18	52	59	111	0.20	0.34	0.11	71	UM7	71	79	152	0.38	0.53	0.17
18	M8	64	33	110	0.01	0.11	0.14	45	M18	98	30	107	0.27	0.22	0.20	72	UM7	82	16	66	0.50	0.50	0.50
19	M9	48	100	70	0.15	0.34	0.42	46	M19	50	12	67	0.31	0.40	0.46	73	UM8	39	75	74	0.03	0.16	0.28
20	M9	67	35	144	0.29	0.14	0.12	47	M19	15	18	83	0.42	0.45	0.44	74	UM8	46	8	41	0.56	0.60	0.65
21	M9	45	60	88	0.04	0.12	0.23	48	M19	40	30	106	0.27	0.28	0.02	75	UM8	12	51	51	0.01	0.03	0.00
22	M10	27	15	60	0.16	0.12	0.26	49	M19	57	15	56	0.46	0.54	0.56	76	UM8	38	40	45	0.44	0.39	0.55
23	M10	40	5	59	0.32	0.35	0.43	50	M20	57	17	53	0.49	0.57	0.58	77	UM9	42	52	85	0.03	0.08	0.21
24	M10	18	32	42	0.29	0.24	0.34	51	M20	22	19	63	0.04	0.03	0.12	78	UM9	24	48	81	0.27	0.37	0.08
25	M11	39	18	60	0.29	0.33	0.41	52	M20	60	24	44	0.59	0.64	0.65	79	UM9	38	63	97	0.27	0.30	0.04
26	M11	82	12	85	0.35	0.37	0.35	53	M20	47	13	59	0.37	0.45	0.50	80	UM9	58	64	118	0.20	0.46	0.07
27	M12	50	13	51	0.47	0.55	0.59	54	M21	54	18	57	0.43	0.52	0.55								
																		MMER			0.29	0.31	0.31

Appendix 3

Names of developers from Federal Commission of Electricity (C*n*) at Guadalajara, Jalisco (Director email: omar.delacruz@cfe.gob.mx); CINVESTAV (P*n*) at Guadalajara, Jalisco (Director email: jesus.vazquez@cts-design.com); Universidad del Valle de Atemajac (U*n*) at Guadalajara, Jalisco (Director email: elena.gonzalez@univa.mx); Guadalajara University (M*n*) at Jalisco (Director email: leonardo.soto@cucea.udg.mx); Universidad del Valle de Atemajac (UM*n*) at Leon, Guanajuato (Director email: luis.garcia@univa.mx).

Verification stage: C1: Barraza A. I., C2: De la Cruz P. O., C3: Flores G. C., C4: Galindo G. R., C5: García R. M., C6: Guerra M. A., C7: Guzmán M. A., C8: Hernández H. P., C9: Hernández R. A., C10: Partida M. L., P1: Alegría B. J., P2: Escamilla R. J., P3: Gutíerrez R. F., P4: Montesinos S. J., P5: Morales L. D., P6: Plascencia S. J., P7: Reynoso R. R., P8: Rivera V. B., P9: Vega B. F., P10: Viramontes C. A., P11: Cordero B. D., P12: Davis A. R., P13: Díaz I. M. J., P14: Domínguez Z. S., P15: Duarte L. M., P16: Jiménez G. N., P17: Montero S.A., P18: Martínez S. N., P19: Rocha H. J., P20: Vega Á. C., P21: González C. D., P22: Gutiérrez R. L., P23: Muñetón P. O., P24: Plata V. P., P25: Tapia G. S., P26: Aguirre Z. M., P27: Calvillo C. C., P28: Gallegos R. L., P29: Hernández O. O., P30: Meza A. E., P31: Ramos C. L., P32: Sapiens P. J., U1: Gutiérrez H. A., U2: Tamayo E., U3: Ayala A. C., U4: Gonzalez Q. R., U5: Navarro N. S., U6: Rivera P. E., U7: Zavala G. H., U8: Cardosa M. T., U9: Cortés F. G., U10: Lugo R. J., U11: Martínez G. O.

Validation stage: M1: Cabral J.J., M2: Dueñas del Toro H. O., M3: González P.J.J., M4: Herrera B. K., M5: Lopez F. A. D., M6: Maciel A. L. A., M7: Moreno G. M., M8: Ramos C. S., M9: Vallejo M. E., M10: Villegas R. M., M11: Carrillo D. I., M12: Castro T. M., M13: Estrada V. L., M14: Garcia V. L., M15: Herrera I. J., M16: Mercado G. S., M17: Peñalba V. A., M18: Ramírez L. E.,

M19: Robledo H. A., M20: Torres A. U., M21: Torres E. A., UM1: Castillo O. R., UM2: Cedillo B. F., UM3: Cruz G. A., UM4: Martínez P. R.,UM5: Padilla H. B., UM6: Palomares A. L., UM7: Ramirez R. M., UM8: Rodríguez S. J., UM9: Zuñiga A. V.

References

1. Ahmed MA, Saliu MO, AlGhamdi J (2005) Adaptive fuzzy logic-based framework for software development effort prediction. Inf Softw Technol 47(1):31–48
2. Boehm B (1981) Software engineering economics. Prentice Hall, Englewood Cliffs
3. Boehm B, Abts Ch, Chulani S (2000) Software development cost estimation approaches: a survey. J Ann Softw Eng 10:177–205
4. Boehm B, Abts Ch, Brown AW, Chulani S, Clarck BK, Horowitz E, Madachy R, Reifer D, Steece B (2000) COCOMO II. Prentice Hall, Englewood Cliffs
5. Boetticher GD, Lokhandwala N (2007) Assessing the reliability of a human estimator. Third international workshop on predictor models in software engineering (PROMISE'07), IEEE Computer Society Press
6. Braz M, Vergilio S (2004) Using fuzzy theory for effort estimation of object-oriented software. In: Proceedings of the 16th IEEE international conference on tools with artificial intelligence, ICTAI
7. Briand LC, Wieczorek I (2001) Software resource estimation. Encyclopedia of software engineering, vol 2. Wiley, New York, pp 1160–1196
8. Briand LC, Langley T, Wieczorek I (2000) A replicated assessment and comparison of common software cost modeling techniques. IEEE international conference on software engineering (ICSE), Limerick, Ireland
9. Brooks FP Jr (2003) Three great challenges for half-century-old computer science. J ACM 50(1):25–26
10. Burguess CJ, Lefley M (2001) Can genetic programming improve software effort estimation? A comparative evaluation. J Inf Softw Technol 43(2001):863–873
11. Conte SD, Dunsmore HE, Shen VY (1986) Software engineering metrics and models. Benjamin/Cummings, M. Park CA
12. De Barcelos TIF, Simies da Silva JD, Sant Anna N (2008) An investigation of artificial neural networks based prediction systems in software project management. J Syst Softw 81(3):356–367
13. Finnie GR, Wittig GE, Desharnais JM (1997) A comparison of software effort estimation techniques: using function points with neural networks, case-based reasoning and regression models. J Syst Softw 39(3):281–289
14. Foss T, Stensrud E, Kitchenham B, Myrtveit I (2003) A simulation study of the model evaluation criterion MMRE. IEEE Trans Softw Eng 29(11):985–995
15. Heiat A (2002) Comparison of artificial neural network and regression models for estimating software development effort. J Inf Softw Technol 44(15):911–922
16. Huang X, Ren J, Capretz LF (2004) A Neuro-Fuzzy tool for software estimation. In: Proceedings of the 20th IEEE international conference on software maintenance
17. Humphrey W (1995) A discipline for software engineering. Addison Wesley, London
18. Humphrey WS (2005) PSP, a self improvement process for software engineers. SEI series in software engineering. Adison Wesley, London
19. Idri A, Abran A, Khoshgoftaar T (2001) Fuzzy analogy: a new approach for software cost estimation. In: International workshop on software measurement, Montréal, QC, Canada
20. Idri A, Abran A, Khoshgoftaar T (2002) Estimating software project effort by analogy based on linguistic values. In: Eight IEEE symposium on software metrics
21. Jørgensen M (2006) A preliminary theory of judgment-based project software effort predictions. In: Ou L, Turner R (eds) IRNOP VIII. Project research conference. Publishing House of Electronic Industry, Beijing, pp 661–668
22. Jørgensen M (2007) A critique of how we measure and interpret the accuracy of software development effort estimation. In: Keung J (ed) 1st International workshop on software productivity analysis and cost estimation. Information Processing Society of Japan, pp 15–22
23. Jørgensen M (2007) Forecasting of software development work effort: evidence on expert judgment and formal models. J Forecast 23(3):449–462
24. Jorgensen M, Kirkeboen G, Sjoberg D, Anda B, Brathall L (2000) Human judgment in effort estimation of software projects. In: International conference on software engineering, Limerick, Ireland
25. Kadoda G, Cartwright M, Chen L, Shepperd M (2000) Experiences using case-based reasoning to predict software project effort. In: Proceedings of the EASE conference keele, UK
26. Khoshgoftaar TM, Allen EB, Xu Z (2000) Predicting testability of program modules using a neural network. In: IEEE symposium on application-specific systems and software engineering technology, pp 57–62
27. Kitchenham BA, Pickard LM, MacDonell SG, Shepperd MJ (2001) What accuracy statistics really measure. In: IEE proceedings software, vol 148(3), pp 81–85
28. Kitchenham BA, Pfleeger SL, Pickard LM, Jones PW, Hoaglin DC, Emam KE, Rosenberg J (2002) Preliminary guidelines for empirical research in software engineering. IEEE Trans Softw Eng 28(8):721–734
29. Kok P, Kitchenhan BA, Kirakowski J (1990) The MERMAID approach to software cost estimation. In: Proceedings ESPRIT
30. Kultur Y, Turhan B, Basar BA (2008) ENNA: software effort estimation using ensemble of neural networks with associative memory. ACM SIGSOFT 2008/FSE-16, November 9–15, Atlanta, Georgia, USA
31. Lewis JP (2001) Large limits to software estimation. ACM Software Engineering Notes, vol 26(4), pp 54–59
32. López-Martín C, Yáñez-Márquez C, Gutiérrez-Tornés A (2008) Predictive accuracy comparison of fuzzy models for software development effort of small programs. J Syst Softw 81(6):949–960. doi:10.1016/j.jss.2007.08.027
33. MacDonell SG (2003) Software source code sizing using fuzzy logic modeling. Elsevier Science, London
34. MacDonell SG, Gray AR (1996) Alternatives to regression models for estimating software projects. In: Proceedings of the IFPUG fall conference, Dallas, TX
35. Mendes E, Mosley N, Watson I (2002) A comparison of case-based reasoning approaches to web hypermedia project cost estimation. In: 8th IEEE international software metrics symposium. IEEE Computer Society, Ottawa, Canada
36. Mie Mie Thet T, Tong-Seng Q (2005) Application of neural networks for software quality prediction using object-oriented metrics. J Syst Softw 76(2):147–156
37. Montgomery D, Peck E (2001) Introduction to linear regression analysis. Wiley, London
38. Musflek P, Pedrycz W, Succi G, Reformat M (2000) Software cost estimation with fuzzy models. Appl Comput Rev 8(2):24–29

39. Park RE (1992) Software size measurement: a framework for counting source statements. Software Engineering Institute, Carnegie Mellon University
40. Park S (2008) An empirical validation of a neural network model for software effort estimation. J Exp Syst Appl 35:929–937
41. Pedrycz W (2002) Computational intelligence as an emerging paradigm of software engineering. ACM **1**4th international conference on Software engineering and Knowledge Engineering
42. Pedrycz W, Gomide F (1998) An introduction to fuzzy sets. The MIT Press, Cambridge
43. Rombach D, Münch J, Ocampo A, Humphrey WS, Burton D (2008) Teaching disciplined software development. J Syst Softw 81:747–763
44. Schofield C (1998) Non-algorithmic effort estimation techniques. ESERG, TR98-01
45. Specht DF (1991) A general regression neural network. IEEE Trans Neural Netw 2(6):568–576
46. Srinivasan K, Fisher D (1995) Machine learning approaches to estimating software development effort. IEEE Trans Softw Eng 21(2):126–137
47. Sun-Jen H, Nan-Hsing Ch, Li-Wei Ch (2008) Integration of the grey relational analysis with genetic algorithm for software effort estimation. J Oper Res 18:898–909
48. Xu Z, Khoshgoftaar TM (2004) Identification of fuzzy models of software cost estimation. Fuzzy Sets Syst 145(1):141–163
49. Zadeh LA (1999) From computing with numbers to computing with words—from manipulation of measurements to manipulation of perceptions. IEEE Trans Circuits Syst I Fundament Theory Appl 45(1):105–119