International Journal of Engineering Education Vol. 32, No. 2(B), pp. 982–994, 2016
Printed in Great Britain

0949-149X/91 $3.00+0.00

A Radial Basis Function Neural Network for Predicting the Effort of Software Projects Individually Developed in Laboratory Learning Environments*

CUAUHTÉMOC LÓPEZ-MARTÍN[1], IVICA KALICHANIN-BALICH[2], ROSA LEONOR ULLOA-CAZAREZ[3] and NOEL GARCÍA-DÍAZ[4]
[1,2,3] Universidad de Guadalajara, CUCEA, Periférico Norte N° 799, Núcleo Universitario Los Belenes, P.O. Box 45100, Zapopan, Jalisco, México.
[4] Universidad de Colima, FIME, P.O. Box 28400, Coquimatlán, Colima, México.
[4] Instituto Tecnológico de Colima, Avenida Tecnológico N° 1, P.O. Box 28976, Villa de Álvarez, Colima, México.
E-mail: [1] cuauhtemoc@cucea.udg.mx, [2] ivicak@cucea.udg.mx, [3] rosi_ulloa@cucea.udg.mx, [4]ngdiaz@ucol.mx

Prediction techniques have been applied for predicting dependent variables related to Higher Education students such as dropout, grades, course selection, and satisfaction. In this research, we propose a prediction technique for predicting the effort of software projects individually developed by graduate students. In accordance with the complexity of a software project, it can be developed among teams, by a team or even at individual level. The teaching and training about development effort prediction of software projects represents a concern in environments related to academy and industry because underprediction causes cost overruns, whereas overprediction often involves missed financial opportunities. Effort prediction techniques of individually developed projects have mainly been based on expert judgment or based on mathematical models. This research proposes the application of a mathematical model termed Radial Basis function Neural Network (RBFNN). The hypothesis to be tested is the following: effort prediction accuracy of a RBFNN is statistically better than that obtained from a Multiple Linear Regression (MLR). The projects were developed by following a disciplined development process in controlled environments. The RBFNN and MLR were trained from a data set of 328 projects developed by 82 students between the years 2005 and 2010, then, the models were tested using a data set of 116 projects developed by 29 students between the years 2011 and first semester of 2012. Results suggest that a RBFNN having as independent variables new and changed code, reused code and programming language experience of students can be used at the 95.0% confidence level for predicting the development effort of individual projects when they have been developed based upon a disciplined process in academic environments.

Keywords: laboratory learning environments; software development effort prediction; radial basis function neural network; multiple linear regression

1. Introduction

The use of computational prediction models has been proposed inside the Engineering Education and Computer Science Learning [1]. These models have been applied in Higher Education Institutions (HEI) by predicting dependent variables regarding students who withdraw before completing a course [2], course enrolling [3], scores on final comprehensive exam [4], student perception in values of satisfactory, neutral and unsatisfactory [5], affective state during an online self-assessment test [6], final grades in a course [7], students who pass or fail the course in terms of the average score for midterm and/or final examinations [8–9], student responds in yes-no questions in a survey (''I am satisfied with this class'', ''This course contributed to my educational development'', ''This course contributed to my professional development''', ''I am satisfied with the level of interaction that happened in this course'', ''In the future, I would be willing to take a fully online course again'') [10], student scores based upon comprehensive basic sciences exam, comprehensive pre-internship exam, and medical school Grade Point Average (GPA) [11], the average of grades obtained by student at each scholar year [12], teacher scores on a summative assessment based on capstone teaching event [13], graduate who is an alumni association member [14], student qualified for the first step of the medical exam after 2 years [15], as well as academic motivation scale involving intrinsic motivations (motivation to know, to accomplish things, to experience stimulation) and extrinsic motivations (external, introjected, and identified regulation) [16].

Software Engineering Education and Training for HEI suggest the application of principles, methods, processes and technology associated with software engineering theory and practices to the development of a software product by a student who individually develops a software project [17]. Individual processes in software development represent a concern for managers because unless software developers have the capabilities provided by individual training, they cannot properly support their teams or consistently and reliably produce

* Accepted 1 September 2015.

quality products [18]. It has implied that several studies have addressed their approach to analyze the performance of students from an individual point of view [19–25]. One of the individual practices in software engineering education and training is the software development effort prediction starting with the development of small projects to be integrated to large systems [17]; actually, effort prediction is one of the three main practices used for training students at personal level, the others two are related to software defects and software size [23].

Creating a highly accurate and reliable model for predicting the development effort of software projects is a continuous concern of researchers and software managers [26] because bad predictions may address to poor planning, low profitability, and, consequently, products with poor quality. At date, researchers have approached their efforts to (1) determine which technique has the greatest effort prediction accuracy, or (2) propose new or combined techniques that could provide better predictions [27–28].

The models used for predicting dependent variables related to students of HEI have been statistical linear regression models (a regression model that contains one [6] or more than one independent variable [2, 7–9, 14, 16]), item response theory model (statistical analysis of measurement scales) [3], support vector machines (a set of related supervised learning methods that analyze data and recognize patterns, used for classification and regression analysis) [4, 8], data mining (exploration of data bases with considerable extension, in hope of finding useful patterns) [5, 10], statistical analysis based on correlations [11, 15, 16] and distributions [11], as well as neural networks (mathematical models that attempt to mimic the connectivity and learning attributes of a human brain) [4, 6, 8, 9, 12, 13].

Those techniques that have been specifically applied for predicting the development effort of software projects could be classified into the following two general categories [29]:

1. Expert judgment. It implies a lack of analytical argumentation and aims to derive predictions based on experience of experts on similar projects; this technique is based on a tacit (intuition-based) quantification step [30].
2. Mathematical models, which are based on a deliberate (mechanical) quantification step [30], such as statistical regression [31], classifiers based on associative memories [32], fuzzy logic [33], mathematical morphology [34] or machine learning such as case-based reasoning, artificial neural networks, decision trees, Bayesian networks, support vector regression, genetic algorithms, genetic programming, and association rules [28].

The techniques that have been specifically applied for predicting the effort of software projects developed individually in academic environments are expert judgment [35], and mathematical models such as fuzzy logic [33, 36], associative memory classifier [32], genetic programming [37], and neural networks as multilayer feedforward perceptron [38] and general regression neural network [39, 40]. Because of statistical regressions are the models most frequently compared with mathematical models [28, 41], the prediction accuracy in the mentioned studies of this paragraph (except in that where expert judgment was applied [35]), was compared with that obtained from statistical regressions.

Regarding neural networks and statistical regressions models, they have also been applied in other fields such as accounting, finance, health, medicine, engineering, manufacturing or marketing. Some problems have been identified when compared accuracies between neural networks and statistical regressions [28, 41]. The identified problems are the following: there have not been clear some of the characteristics like sample size or number of variables, clarity on the determination of parameters of the neural networks, statistical techniques have not been used, results obtained from the trained model are not tested on a new data set that is not used for training the models, and it has not been clear whether statistically significant difference exists in the performance of different techniques that are compared. These problems are taken into account in our study.

The contribution of our study is investigating the application of a mathematical model termed Radial Basis Function Neural Network (hereafter named RBFNN) for predicting the development effort of software projects that have been developed by students graduated from HEI related to computer engineering, such that this RBFNN (1) could be useful for a practitioner by registering his/her own data history following a disciplined process specifically designed for developing software at individual level, and then this obtained data set could be used for generating his/her own RBFNN, and (2) a trainer or teacher could apply the method followed in this study for gathering data of students in several courses and generate his/her own RBFNN for predicting the effort of projects to be developed in future courses. A neural network was selected because it has the ability to learn non-linear functions [42] and non-linear relationships are common among development effort and independent variables in software projects [43]. A RBFNN is pro-

posed once we did not found any study in which a RBFNN has been applied to individually developed software projects. The prediction accuracy of the RBFNN is compared to that of a Multiple Linear Regression (hereafter named MLR).

The RBFNN and MLR were trained from projects developed in a controlled environment and following a disciplined software development process named Personal Software Process (PSP), whose practices and methods have been used in academic environments for delivering quality products on predictable schedule [35] using samples of thousands of students [23]. There is a relationship between PSP and the Capability Maturity Model (CMM). CMM gives an available description of the goals, methods, and practices needed in software engineering industrial practice, while the PSP allows its instrumentation at an individual level: twelve of the eighteen key process areas of the CMM are at least partially considered in PSP [18].

The RBFNN and the MLR are trained using a data set of 328 software projects developed by 82 students between the years 2005 and 2010, then, the models are tested using a new data set of 116 projects developed by 29 students between the years 2011 and first semester of 2012. All of the 111 students were studying a graduate program related to computer science. Graduate students were selected because previous qualitative analyses have shown that within a PSP course, undergraduate students were more concerned with programming than with the software process issues [19, 21].

Three independent variables for predicting the effort were obtained from the projects, two of them related to size measured in source lines of code (*new and changed* code, and *reused* code) and the third one related to people (*programming language experience of students*) reported by the students in months, whereas the dependent variable was measured in minutes. The hypothesis to be investigated in our research is the following:

H_1: Effort prediction accuracy of a RBFNN is statistically better than that obtained from a MLR, when *new and changed code*, *reused code*, and *programming language experience of students* data obtained from projects individually developed with personal practices, are used as independent variables.

The study is organized as follows: Section 2 has been assigned for describing the RBFNN. In Section 3, the independent and dependent variables are defined. Section 4 justifies and defines the prediction accuracy evaluation criterion for the two models. Section 5 is assigned to a detailed analysis of related studies in which models have been proposed for predicting the development effort of individually developed projects. In Section 6 the experimental design followed in order to obtain the data set of software projects is described. Section 7 has been assigned for the results of this study. Subsection 7.1 describes the training and testing of the RBFNN and MLR models. Subsection 7.2 presents the statistical procedure for comparing the prediction accuracy between the two models. Finally, the Section 8 is related to conclusions, limitations and future work of this study.

2. Radial basis function neural network

A RBFNN is a network consisting of one input layer, one middle-layer (hidden layer) and an output layer. A RBFNN has as characteristic that the activation of a hidden unit is related to the distance between the input vector and a prototype vector [42]. These kind of neural networks have their origin in techniques for performing exact interpolation of a set of data points in a multi-dimensional space. The exact interpolation requires every training input vector to be mapped exactly onto the corresponding target vector. The RBFNN included two phases: unsupervised and supervised phases. In its first phase, input data are clustered and cluster details are sent to hidden neurons, where radial basis functions of the inputs are computed by making use of the center and the standard deviation of the clusters. The radial basis functions are similar to kernel functions in kernel regression. The activation function or the kernel function can use a variety of mathematical functions; the following Gaussian radial basis function is the most commonly used [42]:

$$f(x) = e^{\left(-\frac{\|x-c_i\|^2}{2\sigma_i^2}\right)} \quad (1)$$

Where c_i is the center and σ_i is the width of the influence of the i^{th} middle neuron. The double vertical line in both extremes of the numerator means Euclidean distance.

The learning process to adjust the weights of the connections between the hidden layer and the output layer is of supervised learning type, where ordinary least squares technique is used. Thus, the RBFNN involves both unsupervised and supervised learning. Figure 1 shows the architecture of a RBFNN including dependent and independent variables for this study.

The input of the algorithm for the RBFNN is the set of (P, T) pairs, where P is a vector containing project attributes and T is target, or effort. The algorithm also depends on the spread constant termed *spread* for the radial basis layer. The net input to the activation function is the vector dis-

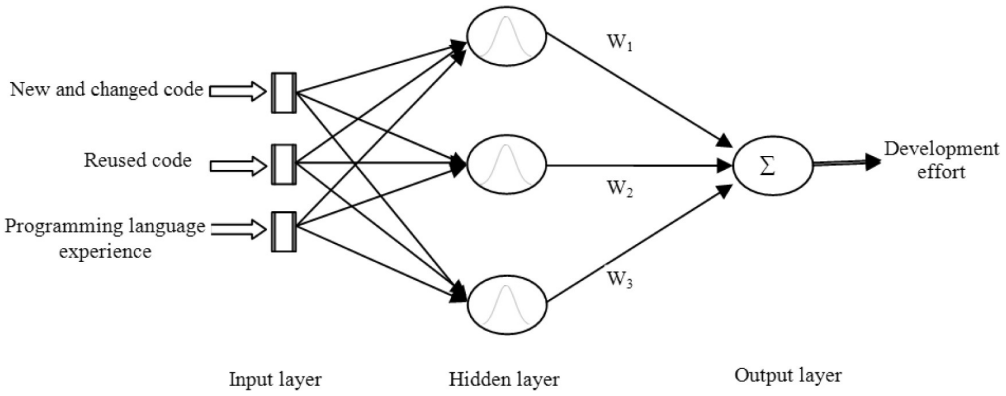

Fig. 1. Architecture of a radial basis function neural network.

tance between its weight vector c_i and the input vector P, multiplied by the bias b. Each neuron in the hidden layer has a bias $b = (0.8326/spread)$. The *spread* gives the distance at which the activation function has value of 0.5. It establishes the width of an area in the input space to which each neuron responds with more than 50% if its value. A smaller *spread* value, a more selective the RBFNN is (i.e., a lower *spread* value, a higher selective RBFNN is). The learning algorithm creates as many radial basis neurons as there are input vectors in the training set, and sets the hidden layer weights to P. That is, there is a layer of radial basis neurons in which each neuron acts as a detector for one distinct input vector. If there are p input vectors, then there will be p neurons.

The learning algorithm returns a network with weights and biases such that the outputs are exactly T when the inputs are P. The output node parameter $\vec{W} = (W_1, W_2, \ldots, W_i, \ldots W_n)$ and B are calculated so that this requirement is fulfilled. It is done by using a trial and solving a linear system as described as follows:

The trial consists of presenting each of the input vectors P from the training set to the RBFNN and memorizing the output vector $\vec{f}$ of the hidden layer as $\vec{f}_i$. Thus, a sequence of vectors $(\vec{f}_1, \vec{f}_2, \ldots, \vec{f}_i, \ldots, \vec{f}_n)$ is created and memorized. Knowing that an input P has to produce an output $T = \theta_i$ the following system of equations can be solved to determine the parameters $\vec{W} = (W, W_2, \ldots, W_i, \ldots W_n)$ and B of the output node:

$$\begin{bmatrix} \vec{f}_1 & 1 \\ \vec{f}_2 & 1 \\ \cdots & \\ \cdots & \\ \cdots & \\ \vec{f}_n & 1 \end{bmatrix} \cdot \begin{bmatrix} W_1 \\ W_2 \\ \cdots \\ \cdots \\ \cdots \\ W_n \\ B \end{bmatrix} = \begin{bmatrix} e_1 \\ e_2 \\ \cdots \\ \cdots \\ \cdots \\ e_n \end{bmatrix}$$

Since no two vectors P are a multiple of each other, the vectors $\vec{f}_i$ are linearly independent; therefore, the system has a unique solution.

3. Software measurement (independent and dependent variables)

Amongst a wide range of software product size measures, source lines of code (LOC) remains in favor of many models [44].

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 [18]. In this research, two of the independent variables are *New and Changed* (N&C) as well as *Reused* code and all of them were 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 project while the reused code is the LOC of previously developed programs that are used without any modification [18]. 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 [18]. Hence, it is necessary to always use the same coding and counting standards. The software projects of this study followed these two guidelines.

After the product size, people factors (such as experience on applications), platforms, languages and tools have the strongest influence in determining the amount of effort required to develop a software product [45]. Therefore, programming language experience is used as a third independent variable in this research, which was measured in months, whereas development effort was measured in minutes.

4. Accuracy criterion

A common criterion used for evaluating the models in the software development effort prediction has been the Mean Magnitude of Relative Error or MMRE [28]; however, it is biased since it is based on ratios which leads to the asymmetry; hence, the use of Absolute Residuals (AR) was recently proposed. AR is unbiased and it is not based on ratios [46]. The AR measure is defined as follows:

$$AR_i = |Actual\ Effort_i - Predicted\ Effort_i| \quad (2)$$

The AR is calculated for each observation i, the effort of which is predicted. The aggregation of the AR over multiple observations (N) can be obtained by the mean (MAR) as follows:

$$MAR = (1/N)\sum_{i=1}^{N} AR_i \quad (3)$$

The accuracy of a prediction model is inversely proportional to the MAR.

5. Related work

Several kinds of neural networks have been applied for predicting the development effort of software projects. In the year of 2008, the feed-forward multilayer perceptron with back propagation learning algorithm was reported as the neural network most commonly used in the effort prediction of large software projects [47], and it has been also used until the year of 2009 [26], 2013 [48] and 2015 [29].

Radial Basis Function Neural Networks have been applied for predicting the software development effort of projects [49–56]; however, the projects they used were developed by teams of developers, in addition, they used the MMRE as accuracy criterion; moreover, they did not report statistical significance nor checked the assumptions for dependence and normality data. We have not found any study in which a RBFNN is applied to individually developed projects. Table 1 describes those identified studies where the development effort of individually developed projects has been predicted using a model. The prediction accuracy of all the models proposed in Table 1 was compared to that of a statistical regression. Results showed that in the testing phase of models, the accuracies of predictions achieved by fuzzy logic [33, 36], classifier based on associative memory [32], multilayer neural network [38], genetic programming [37], as well as a general regression neural network [39, 40], were statistically equivalent to a least squares regression model (either simple or MLR).

In our study, *new and changed* code, *reused* code and *programming language experience of students* are used as independent variables for training and testing the RBFNN and the MLR models; the sample sizes for training and for testing the models were: 328 for training (developed between the years 2005–2010 by 82 students) and 116 for testing (developed in the year 2011 and first semester of 2012 by 29 students).

6. Experimental design

In our study, the experiment used for obtaining the data has already been used in all those studies included in Table 1. The experiment was done within a controlled environment having the following characteristics:

1. All of the students were experienced and were working for some software development enterprise. However, none of them had previously taken a course related to personal practices for developing software at the individual level.
2. The kind of the developed projects had a similar complexity as those suggested in [18]. From a set of 18 projects, a subset of seven projects was randomly assigned to each of the students. Description of these 18 projects is presented in [35].
3. Each student developed seven project assignments. Only the last four of the assignments of each developer were selected for this study. The first three projects were not considered because they had differences in their process phases and

Table 1. Studies of individually developed software projects (N&C: New and changed lines of code; Reused: Reused lines of code; PLE: Programming language experience of students reported in months; NSP: Number of software projects; NS: Number of students who developed the projects; PT: Period of time the projects were developed)

Study	Proposed model	Independent variable(s)	Training			Testing		
			NSP	NS	PT	NSP	NS	PT
[33]	Fuzzy logic	N&C	105	30	2005–2006	20	7	2007
[36]	Fuzzy logic	N&C and Reused	163	53	2005–2007	68	21	2008
[32]	Associative memory	N&C and Reused	163	53	2005–2007	68	21	2008
[38]	Multilayer neural network	N&C and Reused	132	40	2005–2008	77	24	2009
[39]	General regression neural network	N&C and Reused	163	53	2005–2007	80	30	2008–2009
[40]	General regression neural network	N&C, Reused and PLE	156	51	2005–2008	156	47	2009–2010
[37]	Genetic programming	N&C, Reused and PLE	219	71	2005–2009	130	38	2010

in their logs, whereas the last four projects were based on the same logs and they had the following phases [18]: (1) Plan: plan the work and document it by predicting the size and development effort of projects; (2) Design: represent the program either by using a flow diagram or pseudo-code; (3) Design review: personal reviews conducted by a student on his/her own previous design errors; (4) Code: implement the design by using a programming language; (5) Code review: personal reviews conducted by a student on his/her own previous code errors; (6) Compile: compile the program and fix and log all defects found; (7) Testing: execute the compiled program and fix and log all defects found; and (8) Postmortem: record actual size, development effort, and defect data on the plan. Design and code reviews were structured, they involved data-driven review processes that were guided by personal review checklists derived from the historical defect data of the student. As for logs used by student were [18]: (1) Plan Summary Form recording planned and actual data regarding size and development effort by phase, as well as defects; (2) Time recording log including the date of the assignment; start and finish times by phase; (3) Defect recording log involving the date of the documented error, number and type of error, phase in which the error was injected, phase in which the error was removed, fix time, and description of the error; (4) Process improvement proposal which provided students to record process problems and proposed solutions.

4. Each developer selected his/her own imperative programming language whose coding standard had the following characteristics: each compiler directive, variable declaration, constant definition, delimiter, assign sentence, as well as flow control statement was written in one line of code.
5. Students had already received at least one formal course on the programming language that they selected to be used through the assignments. Because the type of programming language is a main factor that has significant influence on the productivity of developers [57], in this study the selected projects were those coded using programming languages of a same paradigm (object oriented) and same generation (the third). The sample for this study involved only those students whose projects were coded in C++ or JAVA.
6. Because this study was an experiment, with the aim of reducing bias the students were not informed about the experimental goal.

Table 2. Counting standard

(1) Count type: Physical	
(2) Statement type	Included? (Yes/No)
(2a) Executable	Yes
(2b) No executable	
Declarations	Yes, one by text line
Compiler directives	Yes, one by text line
Comments and Blank lines	No
(3) Clarifications	
{ and }	Yes

7. Students filled out a spreadsheet for each project and submitted it electronically for examination and feedback.
8. Each course was taught to no more than fifteen students.
9. Since a coding standard establishes a consistent set of coding practices that is used as a criterion for judging the quality of the produced code [18], the same coding and counting standards were used in all projects. The projects developed during this study followed these guidelines. All projects adhered to the counting standard shown in Table 2.
10. Students were constantly supervised and advised about their process. Data used in this study are from those students whose data for all seven assignments were correct, complete, and consistent.
11. The code written in each project was designed by the developers to be reused in subsequent projects.

7. Results

7.1 Training and testing the models

The following MLR equation was generated [58] from the actual data of the 328 projects developed between the years 2005–2010 by 82 students (Appendix A):

$$Effort = 63.735 + (1.037 * N\&C) - (0.186 * Reused) - (0.425 * Programming\ Language\ Experience) \quad (4)$$

In accordance with the theoretically justifiable functional form, it is first necessary to analyze if the predicted effort relates correctly to the data of its independent variables [59], that is, if the relationship between *effort* and *new and changed code*, *reused code*, and *programming language experience of students* is according to the software development theory. According to the signs of each of the three parameters of the MLR, the model meets with the following assumptions related to the software development:

1. The higher the value of *new and changed code* (N&C), the higher the *development effort* is.

2. The higher the value of *reused code*, the lower the *development effort* is.
3. The higher the value of *programming language experience of students*, the lower the *development effort* is.

When a model is used for predicting development effort, an acceptable value for the coefficient of determination is $r^2 \geq 0.5$ [18]. Equation 4 had a r^2 equal to 0.51. The ANOVA for this equation had a statistically significant relationship among the variables at the 99% confidence level. To determine whether the model could be simplified, a parameter analysis for the MLR was done. Results of this analysis showed that the highest p-value of the three independent variables was 0.0002, which corresponded to *reused code*. Since this p-value was less than 0.01, *reused code* is statistically significant at the 99% confidence level and consequently, the independent variable of *reused code* was not removed.

The MLR equation was then applied to the original data set of 328 projects, and an AR for each project, as well as a MAR for the MLR, was then calculated. The MLR yielded a MAR = 17.51.

Regarding the RBFNN, to determine its *spread* value, a *k-fold cross validation* was used on the training set. A $k = 10$ was selected since it is the most common value in the machine learning field [60]. The lower (best) MAR for the RBFNN was equal to 16.52; it resulted when the *spread* value was equal to 570. A spread value which is either lower or higher than *spread* = 570 generated a higher value than MAR = 16.52 for the training set.

In the testing phase, the trained RBFNN and MLR were applied to a new data set of 116 projects developed in the year 2011 and first semester of 2012 by 29 students, whose data were obtained from experiments that had the same characteristics as those projects used for training the models (Appendix B). The MLR yielded a MAR = 19; whereas the MAR for the RBFNN had a value of 17.88.

7.2 Accuracy comparison between models

The prediction accuracy of the MLR and RBFNN models should be statistically compared [61]. The selection of a suitable statistical test for comparing of the accuracy of the two models should be taking into account the assumptions of dependence and normality [62]:

(a) Dependence: Data of the software projects can be described as n pairs (X_i, Y_i), $i = 1, \ldots, n$, where i is the *i-th* project, n is the number of projects, X_i and Y_i are the ARs obtained from the MLR and RBFNN models, respectively. Since each of the pairs X_i and Y_i is obtained from the corresponding project i, then $X_i, \ldots, X_n$ and $Y_i, \ldots, Y_n$, are pairwise dependent samples; therefore a procedure to test the differences between the two sets X_i, and Y_i (MLR – RBFNN) should be selected for determining whether the MAR set of 116 differences is equal or greater than zero with statistically significant difference. If a MAR should be used for comparing the accuracy between models, a normality test for the set of those 116 differences should be applied.
(b) Normality: The tests for normality of a data set are Chi-squared, Shapiro-Wilk, skewness, and kurtosis.

Table 3. Statistical tests for data normality (testing phase)

	MLR–RBFNN
Chi-Squared	0.0045
Shapiro-Wilk	0.0000
Skewness	0.0000
Kurtosis	0.0000

Table 3 shows the results of the four normality tests on MLR – RBFNN data set in testing phase. Since the smallest p-value amongst some of the tests performed is less than 0.01, we can reject with 99% confidence that MLR – RBFNN data set comes from a normal distribution.

Based upon the finding that the two data sets of this study are dependent and not normally distributed, the suitable statistical test for comparing the accuracies of the two models may be the Wilcoxon test [63]. After applying this statistical test, a p-value equal to 0.0489 was obtained, that is, there was a statistically significant difference amongst the medians of the MLR and the RBFNN at the 95.0% confidence level.

8. Discussion

Several dependent variables regarding Higher Education students have been predicted by applying techniques based upon mathematical models. These variables have been dropout [2], course selection [3], academic performance [4, 9, 11, 12], course satisfaction [5, 10], mood [6], achievement [7], learning performance [8], teacher performance from a student point of view [13], alumni association membership [14], academic success [15], and motivation [16]. Based upon the results that no single technique prediction is best for all situations, and that a careful comparison of the results of several approaches is most likely to produce realistic predictions [66], in our study, a RBFNN was proposed for predicting the development effort of software projects.

Owning to the fact that the levels of software projects in engineering education and training have

been classified in the small and in the large projects [17], previous studies on development effort prediction have addressed their approach by proposing models for predicting the development effort of software projects separately for each level. Software projects have been developed either by teams or individually developed by practitioners as those included in Table 1. The approach of studies related to individual projects is based upon the following assumptions: the productivity of people is primarily influenced by the variations of average team size for the development [64]; the performance of a software 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 members is, at least in part, determined by the practices these engineers follow in doing their work [23]. That is, people productivity differs when they develop software alone or in team, and the performance on individual level is important because of its influence on team performance. Therefore, the analysis of individually developed software projects has been an ongoing concern [19–25].

In our study, each problem identified in [28, 41] when comparing neural networks with statistical regressions was also addressed as follows:

1. The sizes of samples as well as their dependent and independent variables were described with detail.
2. Two independent data samples of software projects were used, one of them for training the models and other sample for testing them. The two mentioned samples were developed by following the same experiment design. Dependent variable was the development effort, whereas independent variables were related to size of projects and with people factors.
3. The RBFNN contained a parameter named *spread* that had influence in the RBFNN prediction accuracy. The manner how this final parameter was determined, was mentioned in this study.
4. Statistical techniques were used to select the significant variables as has been suggested [41] [65] based upon a regression analysis, which was made from a global analysis (based on coefficient of determination) as well as from an individual statistical analysis of its parameters. In addition, a functional form of the statistical regression (MLR) was also analyzed as suggested in [59]. Finally, once the assumptions of dependence and normality of the results of evaluation criterion by model (absolute residuals) were done, the prediction accuracy comparison of two models was based upon a suitable statistical test.

Regarding studies of software engineering in the context of education and training, we identified the following ones related to this study:

1. A comparison between teaching *agile* and *disciplined* processes suggested that a more efficient approach for inexperienced students in software engineering would be a disciplined process [67], which means that the process used in our study was suitable since that the PSP is a disciplined process, and because all the participants did not have any experience in a discipline process, once they had not received any PSP course.
2. A common concern identified in students is how they solve problems during the software development project [68], in the process used in our study, a *process improvement proposal (PIP)* was used for all the students (a *PIP* provides students to record process problems and proposed solutions).
3. Software verification and validation (V&V) is one of the significant areas of software engineering. Therefore, a study covered static V&V in a educational environment [69]. To achieve a static verification and validation activities, our study included design and code review phases by using design and code review checklists obtained from defect recording logs (derived from the historical defect data by student).

9. Conclusions, limitations and future work

After the RBFNN proposed for predicting the development effort of software projects that have been individually developed for graduate students in a controlled laboratory learning environment was trained and tested, the following hypothesis was accepted:

H_1: Effort prediction accuracy of a RBFNN is statistically better than that obtained from a MLR, when *new and changed code*, *reused code*, and *programming language experience of students* data obtained from projects individually developed with personal practices are used as independent variables.

In comparing our study with those ones described in Table 1, the models proposed in each of the studies described in Table 1 reported prediction accuracy statistically equivalent to its corresponding statistical regression model, whereas in our study the RBFNN had prediction accuracy statistically better than that of a MLR at the 95.0%

confidence level. It means that the RBFNN outperformed the prediction accuracy obtained from proposed models such as fuzzy logic, associative memories, genetic programming and two kinds of neural networks: multilayer feedforward perceptron and general regression neural network.

Reliability of a prediction model depends on the quality of data from which the model has been generated: if the data were obtained from a source without any control or having a wide variability in its environment such as following different processes of development, recording the data of projects in a variety of logs, the use of various coding or counting standards, or coded projects on programming languages of several generations, then its accuracy could be uncertain. This fact should not only be related to a RBFNN, but for any other prediction model.

As for the limitations of our study, in spite that the sample size was bigger than similar previous studies, it only involved two kinds of imperative programming languages (C++ and Java); in the future it would be interesting to propose additional models involving projects from other kind of programming paradigms (such as declarative). In addition, more factors should be involved, such as those related to people (as development experience of students), platforms, and software tools.

Results of our research suggest that a RBFNN can be used for predicting the development effort of projects when they have been developed individually in a controlled environment and based upon a disciplined process as that suggested by the Personal Software Process. This conclusion was based on the use of a bigger sample data set which included more recent software projects than all previous studies described in Table 1.

A practical implication of our study is that RBFNN could be useful for a practitioner who registers his/her own data history following a disciplined process. Then, from this data set, the practitioner could generate models such as MLR or RBFNN for predicting the effort of his individually developed projects. In addition, a trainer or teacher could apply the method used in this study by gathering data of student performance in several courses and generate his/her own models for predicting the effort of projects to be developed in future courses or to be added to large software systems.

Future research involves the use of support vector regression machines for predicting the development effort of software projects developed either individually or by teams of students.

Acknowledgments—The authors would like to thank the CUCEA of Universidad de Guadalajara, Universidad de Colima, Instituto Tecnológico de Colima, and Consejo Nacional de Ciencia y Tecnología (CONACyT) for their support during the development of this research.

References

1. C. Yañez-Márquez, M. Aldape Pérez, I. López Yáñez and O. Camacho Nieto, Emerging Computational Tools: Impact on Engineering Education and Computer Science Learning, *International Journal of Engineering Education*, **30**(2), 2014, pp. 1–10.
2. I. Lykourentzou, V. Nikolopoulos, G. Mpardis and V. Loumos, Dropout prediction in e-learning courses through the combination of machine learning techniques, *Computers & Education*, *Elsevier*, **53**(3), 2009, pp. 950–965. DOI: 10.1016/j.compedu.2009.05.010.
3. A. Kardan, H. Sadeghi, S. S. Ghidary and M. R. Fani Sani, Prediction of students course selection in online higher education institutes using neural network, *Computers & Education*, *Elsevier*, **65**, 2013, pp. 1–11. DOI:10.1016/j.compedu.2013.01.015.
4. S. Huang and N. Fang, Predicting student academic performance in an engineering dynamics course: a comparison of four types of predictive mathematical models, *Computers & Education*, *Elsevier*, **61**, 2013, pp. 133–145. DOI:10.1016/j.compedu.2012.08.015.
5. W. W. Guo, Incorporating statistical and neural network approaches for student course satisfaction analysis and prediction, *Expert Systems with Applications, Elsevier,* **37**(4), 2010, pp. 3358–3365. DOI:10.1016/j.eswa.2009.10.014
6. C. N. Moridis and A. A. Economides, Prediction of students mood during an online test using formula-based and neural network-based method, *Computers & Education*, *Elsevier*, **53**, 2009, pp. 644-652. DOI:10.1016/j.compedu.2004.04.002.
7. I. Lykourentzou, I. Giannoukos, G. Mpardis, V. Nikolopoulos and V. Loumos, Early and dynamic student achievement prediction in e-learning courses using neural networks, *Journal of the American Society for Information Science and Technology, Wiley Online Library*, **60**(2), 2008, pp. 372–380. DOI: 10.1002/asi.20970.
8. Y. Hu, C. Lo and S. Shih, Developing early warning systems to predict student's online learning performance, *Computers in Human Behavior, Elsevier*, **36**, 2014, pp. 469-478. DOI: 10.1016/j.chb.2014.04.002.
9. C. Romero, M. I. López, J. M. Luna and S. Ventura, Predicting students' final performance from participation in on–line discussion forums, *Computers & Education*, *Elsevier*, **68**, 2013, pp. 458-472. DOI:10.1016/j.compedu.2013.06.009.
10. Y. Kuo, A. E. Walker, K. E. E. Schroder and B. R. Belland, Interaction, Internet self-efficacy, and self-regulated learning as predictors of student satisfaction in online education courses, *Internet and Higher Education, Elsevier*, **20**, 2014, pp. 35-50, DOI: 10.1016/j.iheduc.2013.10.001.
11. Y. Farrokhi-Kahejh-Pasha, S. Nedjat, A. Mohammadi, E. M. Rad, R. Majdzadeh, F. Monajemi, E. Jamali and S. Yazdani, The validity of Iran's national university entrance examination (Konkoor) for predicting medical students' academic performance, *BMC Medical education, BioMed Central,* **12**(60), 2012, pp. n/d. DOI: 10.1186/1472-6920-12-60.
12. P. Poole, B. Shulruf, J. Rudland and T. Wilkinson, Comparison of UMAT scores and GPA in prediction of performance in medical school: a national study, *Medical Education, Wiley Online Library*, **46**(2), 2012, pp. 163–171. DOI: 10.1111/j.1365.2923.2011.04078.x.
13. J. H. Sandholtz and L. M. Shea, Predicting performance. A comparison of university supervisors' predictions and teacher candidates' scores on a teaching performance assessment, *Journal of Teacher Education, SAGE*, **63**(1), pp. 39–50, DOI: 10.1177/0022487111421175.
14. M. D. Newman, J. M. Petrosko, Predictors of alumni association membership, *Research in Higher Education,*

Springer Link, **52**(7), pp. 738-759. DOI: 10.1007/s11162-011-9213-8.
15. J. C. Hissbach, D. Klusmann, W. Hampe, Dimensionality and predictive validaty of the HAM.Nat, a test of natural sciences for medical school admission, *BMC Medical Education, BIOMed Central*, **11**(83), pp.n/d. DOI: 10.1186/1472/-6920-11-83.
16. M. Muñoz-Organero, P.J. Muñoz-Merino and C. Delgado, Student behavior and interaction patterns with an LMS as motivation predictors in e-learning settings, *IEEE Transactions on Education, IEEE*, **53**(3), pp. 463–470. DOI: 10.1109/TE.2009.2027433.
17. D. J. Bagert, T. B. Hilburn, G. Hislop, M. Lutz, M. McCracken and S. Mengel, *Guidelines for Software Engineering Education Version 1.0*. CMU/SEI-99-TR-032, ESC-TR-99-002, Software Engineering Institute, Carnegie Mellon University, 1999.
18. W. Humphrey *A Discipline for Software Engineering*. Addison Wesley, 1995.
19. S. K. Lisack, The Personal Software Process in the Classroom: Student Reactions (An Experience Report). *13th IEEE Conference on Software Engineering Education & Training*, Austin, Texas, USA, March 6–8, 2000. DOI: 10.1109/CSEE.2000.827035.
20. L. Prechelt and B. Unger, An experiment measuring the effects of PSP Training. *IEEE Transactions on Software Engineering*, **27**(5), pp. 465–472, 2000. DOI: 10.1109/32.922716.
21. P. Runeson, Experiences from Teaching PSP for Freshmen. *14th IEEE Conference on Software Engineering Education & Training*, Charlotte, North Carolina, USA, Feb 9–21, 2001, pp. 98–107. DOI: 10.1109/CSEE.2001.913826.
22. C. Wohlin, Are individual differences in software development performance possible to capture using a quantitative survey? *Empirical Software Engineering, Springer*, **9**(3), 2004, pp. 211–228. DOI: 10.1023/B:EMSE.0000027780.08194.b0.
23. D. Rombach, J. Münch, A. Ocampo, W. S. Humphrey and D. Burton, Teaching disciplined software development. *Journal Systems and Software, Elsevier*, **81**(5), 2008, pp. 747–763. DOI:10.1016/j.jss.2007.06.004.
24. [C. F. Kemerer and M. C. Paulk, The impact of design and code reviews on software quality: an empirical study based on PSP data. *IEEE Transactions on Software Engineering*, **35**(4), 2009, pp. 534–550. DOI: 10.1109/TSE.2009.27.
25. W. H. Shen, N. L. Hsueh and W. M. Lee, Assessing PSP effect in training disciplined software development: A Plan–Track–Review model, *Information and Software Technology, Elsevier*, **53**(2), 2011, pp. 137–148. DOI: 10.1016/j.infsof.2010.09.004.
26. S. Berlin, T. Raz, C. Glezer and M. Zviran, Comparison of estimation methods of cost and duration in IT projects, *Information and Software Technology, Elsevier*, **51**, 2009, pp. 738–748. DOI: 10.1016/j.infsof.2008.09.007.
27. M. Jørgensen and M. J. Shepperd, A Systematic Review of Software Development Cost Estimation Studies, *IEEE Transactions Software Engineering*, **33**(1), 2007, pp. 33–53. DOI: 10.1109/TSE.2007.256943.
28. [J. Wen, S. Li, Z. Lin, Y. Hu and C. Huang, Systematic literature review of machine learning based software development effort estimation models. *Information and Software Technology, Elsevier*, **54**(1), 2012, pp. 41–59. DOI: 10.1016/j.infsof.2011.09.002.
29. C. López-Martín, Predictive accuracy comparison between neural networks and statistical regression for development effort of software projects, *Applied Soft Computing, Elsevier*, **27**, 2015, pp. 434-449. DOI: 10.1016/j.asoc.2014.10.033.
30. T. Halkjelsvik and M. Jørgensen, From origami to software development: A review of studies on judgment-based predictions of performance time, *Psychological Bulletin*, **138**(2), 2012, pp. 238–271. DOI: 10.1037/a0025996.
31. Y. Yang, Z. He, K. Mao, Q. Li, V. Nguyen, B. Boehm and R. Valerdi, Analyzing and handling local bias for calibrating parametric cost estimation models, *Information and Software Technology, Elsevier*, **55**(8), 2013, pp. 1496–1511. DOI: 10.1016/j.infsof.2013.03.002.
32. C. López-Martín, I. López-Yáñez and C. Yáñez-Márquez, Application of Gamma Classifier to Development Effort Prediction of Software Projects, *Applied Mathematics & Information Sciences (AMIS)*, *Natural Sciences Publishing Corporation*, **6**(3), 2012, pp. 411–418.
33. C. López-Martín, C. Yañez-Marquez and A. Gutierrez-Tornes, Predictive Accuracy Comparison of Fuzzy Models for Software Development Effort of Small Programs, *Journal of Systems and Software, Elsevier*, **81**(6), 2008, pp. 949–960. DOI: 10.1016/j.jss.2007.08.027.
34. R. A. Araújo, A. L. I. Oliveira, S. Soares and S. Meira, An evolutionary morphological approach for software development cost estimation, *Neural Networks, Elsevier*, **32**, 2012, pp. 285–291. DOI: 10.1016/j.neunet.2012.02.040.
35. C. López-Martín and A. Abran, Applying Expert Judgment to Improve an Individual's Ability to Predict Software Development Effort, *International Journal of Software Engineering and Knowledge Engineering (IJSEKE), World Scientific,* **22**(4), 2012, pp. 467-483. DOI: 10.1142/S0218194012500118.
36. C. López-Martín, A fuzzy logic model for predicting the development effort of short scale programs based upon two independent variables, *Applied Soft Computing, Elsevier*, **1**(1), 2011, pp. 724–732. DOI: 10.1016/j.asoc.2009.12.034.
37. A. Chavoya, C. Lopez-Martin, R. Andalon and M. E. Meda, Genetic programming as alternative for predicting development effort of individual software projects, *PLOS ONE Journal*, **7**(11), 2012, e50531. DOI: 10.1371/journal.pone.0050531.
38. I. Kalichanin-Balich and C. Lopez-Martin, Applying a Feedforward Neural Network for Predicting Software Development Effort of Short-Scale Projects. *IEEE Software Engineering Research and Applications (SERA)*, Montreal, Canada, 2010. DOI: 10.1109/SERA.2010.41.
39. C. López-Martín, Applying a general regression neural network for predicting development effort of short-scale programs. *Neural Computing and Applications, Springer-Verlag*, **20**(3), 2011, pp. 389-401. DOI: 10.1007/s00521-010-0405-5.
40. C. López-Martín, A. Chavoya and M. E. Meda, Software Development Effort Estimation in Academic Environments Applying a General Regression Neural Network Involving Size and People Factors, *3rd Mexican Conference on Pattern Recognition, LNCS 6718, Springer-Verlag*, Cancun, Mexico, 2011, pp. 269–277. DOI: 10.1007/978-3-642-21587-2_29.
41. M. Paliwal and U. A. Kumar, Neural networks and statistical techniques: A review of applications, *Expert Systems with Applications, Elsevier*, **36**(1), 2009, pp. 2–17. DOI: 10.1016/j.eswa.2007.10.005.
42. C. M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, 1995.
43. H. Chao-Jung and H. Chin-Yu, Comparison of weighted grey relational analysis for software effort estimation. *Software Quality Journal, Springer-Verlag*, **19**(1), 2011, pp. 165–200. DOI: 10.1007/s11219-010-9110-y.
44. S. D. Sheetz, D. Henderson and L. Wallace, Understanding developer and manager perceptions of function points and source lines of code, *The Journal of Systems and Software, Elsevier*, **82**(9), 2009, pp. 1540–1549. doi: 10.1016/j.jss.2009.04.038.
45. B. Boehm, C. Abts, A. W. Brown, S. Chulani, B. K. Clarck, E. Horowitz, R. Madachy, D. Reifer and B. Steece, *COCOMO II*, Prentice Hall, 2000.
46. M. Shepperd and S. MacDonel, Evaluating prediction systems in software project estimation, *Information and Software Technology, Elsevier*, **54**(8), 2012, pp. 820–827. DOI:10.1016/j.infsof.2011.12.008.
47. H. Park and S. Baek, An empirical validation of a neural network model for software effort estimation, *Expert Systems with Applications, Elsevier*, **35**(3), 2008, pp. 929–937. DOI: 10.1016/j.eswa.2007.08.001.
48. A. B. Nassif, D. Ho and L. F. Capretz, Towards an early software estimation using log-linear regression and a multilayer perceptron model, *The Journal of Systems and Software, Elsevier*, **86**(1), 2013, pp. 144–160. DOI: 10.1016/j.jss.2012.07.050.
49. M. Shin and A. L. Goel, Empirical Data Modeling in Software Engineering Using Radial Basis Functions, *IEEE*

Transactions on Software Engineering, **26**(6), 2000, pp. 567–576. DOI: 10.1109/32.852743.
50. A. Heiat, Comparison of artificial neural network and regression models for estimating software development effort. *Information and Software Technology, Elsevier*, **44**(15), 2002, pp. 911-922. DOI: 10.1016/S0950-5849(02) 00128-3.
51. A. L. I. Oliveira, Estimation of software project effort with support vector regression, *Neurocomputing, Elsevier*, **69**, 2006, pp. 1749–1753. DOI: 10.1016/j.neucom.2005.12.119.
52. A. Idri, A. Zahi, E. Mendes and A. Zakrani, Software Cost Estimation Models Using Radial Basis Function Neural Networks, *IWSM-Mensura*, LNCS 4895, Palma de Mallorca, Spain, November 5-8, 2008, pp 21–31. DOI: 10.1007/978-3-540-85553-8_2.
53. K. V. Kumar, V. Ravi, M. Carr and N. R. Kiran, Software development cost estimation using wavelet neural networks, *The Journal of Systems and Software, Elsevier*, **81**, 2008, pp. 1853–1867. DOI: 10.1016/j.jss.2007.12.793.
54. C. S. Reddy, P. S. Rao, K. Raju and V. V. Kumari, A New Approach For Estimating Software Effort Using RBFN Network, *International Journal of Computer Science and Network Security*, **8**(7), 2008, pp. 237–241.
55. A. Idri, A. Zakrani and A. Zahi, Design of Radial Basis Function Neural Networks for Software Effort Estimation, *International Journal of Computer Science*, **7**(4), 2010, pp. 11–17.
56. P. V. G. D. Prasad Reddy, K. R. Sudha, P. Rama-Sree and S. N. S. V. S. C. Ramesh, Software Effort Estimation using Radial Basis and Generalized Regression Neural Networks. *Journal of Computing*, **2**(5), 2010, pp. 87–92.
57. Z. Jiang and C. Comstock, The Factors Significant to Software Development Productivity. *International Journal of Computer, Information, Mechatronics, Systems Science and Engineering, World Academy of Science, Engineering and Technology*, **1**, 2007, pp. 160–164.
58. D. Montgomery and E. Peck, *Introduction to Linear Regression Analysis*, John Wiley, 2001.
59. I. Myrtveit and E. Stensrud, Validity and reliability of evaluation procedures in comparative studies of effort prediction models, *Empirical Software Engineering, Springer*, **17**, 2012, pp. 23-33. DOI: 10.1007/s10664-011-9183-7.
60. P. Refaeilzadeh, L. Tang and H. Liu, Cross-Validation, *Encyclopedia of Database Systems*, Springer-Verlag Berlin Heidelberg. 2009. DOI: 10.1007/SpringerReference_63669.
61. B. A. Kitchenham and E. Mendes, Why comparative effort prediction studies may be invalid. *IEEE 5th International Conference on Predictor Models in Software Engineering, PROMISE*, Vancouver, Canada, May 18–19, 2009. DOI: 10.1145/1540438.1540444.
62. S. M. Ross, *Introduction to Probability and Statistics for Engineers and Scientists*, Third Edition, Elsevier Press, 2004.
63. W. J. Conover, *Practical Nonparametric Statistics*, Third Edition, Wiley, 1999.
64. J. Zhizhong, P. Naudé and C. Comstock, An Investigation on the Variation of Software Development Productivity, *International Journal of Computer, Information, Mechatronics, Systems Science and Engineering, World Academy of Science, Engineering and Technology,* **1**, 2007, pp. 72–81.
65. B. A. Kitchenham, E. Mendes E. and G. H. Travassos, Cross versus Within-Company Cost Estimation Studies: A Systematic Review, *IEEE Transactions on Software Engineering*, **33**(5), 2007, pp. 316–329. DOI: 10.1109/TSE.2007.1001.
66. B. Boehm, C. Abts and S. Chulani, Software development cost estimation approaches: A survey, *Journal of Annals of Software Engineering, Springer*, **10**, 2000, pp. 177–205. DOI: 10.1023/A:1018991717352.
67. P. N. Robillard and M. Dulipovici, Teaching Agile versus Disciplined processes, *The International Journal of Engineering Education*, **24**(4), 2008, pp. 671–680.
68. J. Vanhanen and T. O. A. Lehtinen, Software Engineering Problems Encountered by Capstone Project Teams, *The International Journal of Engineering Education*, **30**(6A), 2014, pp. 1461–1475.
69. D. Mishra, T. Hacaloglu and A. Mishra, Teaching Software Verification and Validation Course: A Case Study, *The International Journal of Engineering Education*, **30**(6A), 2014, pp. 1476–1485.

Appendix A.

Data set of 328 software projects developed by 82 students between the years 2005 and 2010 used for training the models. N&C: New and changed lines of code; R: Reused lines of code; PLE: Programming language experience of students (in months); E: Effort (in minutes).

N&C	R	PLE	E	N&C	R	PLE	E	N&C	R	PLE	E	N&C	R	PLE	E
17	21	24	65	41	12	6	71	20	30	12	88	35	64	36	67
36	24	24	100	18	39	48	60	49	19	12	95	54	40	36	130
42	11	24	100	43	8	48	119	52	25	12	84	17	38	6	95
95	8	14	171	56	12	48	87	24	81	36	73	64	23	6	110
97	30	14	131	18	108	12	34	25	46	36	53	48	10	6	128
34	78	24	43	36	40	12	81	37	25	36	85	20	54	18	59
35	38	24	52	13	50	12	51	52	59	36	75	31	21	18	71
50	80	24	52	13	48	12	58	27	70	12	97	34	61	18	86
11	33	12	44	17	43	12	89	45	19	12	102	50	66	18	93
25	8	12	50	30	25	12	71	50	43	12	104	10	42	36	83
10	42	36	35	54	94	8	100	68	81	12	115	16	47	36	45
12	25	36	38	8	64	12	53	18	23	12	87	27	25	36	83
23	22	36	54	22	30	12	84	24	52	12	112	38	30	36	121
31	33	36	61	29	40	12	79	31	77	12	57	21	52	36	64
15	44	12	41	40	19	12	68	52	70	12	130	25	30	36	56
22	23	12	41	16	71	12	77	25	77	36	61	44	52	36	69
24	38	12	39	37	71	12	74	35	40	36	81	53	50	36	111
40	25	60	112	40	31	12	122	35	58	36	100	11	43	12	80
14	31	36	79	71	39	12	177	12	47	12	92	22	33	12	63
31	33	36	85	19	70	36	35	32	5	12	98	34	37	12	113
35	31	36	107	32	74	36	82	36	10	12	82	35	53	36	48
18	40	24	70	66	21	36	84	45	19	12	128	43	23	36	52
18	33	24	97	29	104	6	85	15	100	36	29	40	17	36	90
46	10	48	83	58	9	6	99	21	53	36	45	20	31	60	64
81	32	24	155	59	36	6	85	13	57	12	76	32	36	60	60
119	100	24	168	67	6	6	97	22	35	12	107	47	51	60	112
18	44	24	32	44	30	60	112	33	24	12	126	55	56	60	72

N&C	R	PLE	E	N&C	R	PLE	E	N&C	R	PLE	E	N&C	R	PLE	E
37	12	24	50	41	100	36	58	8	30	48	38	17	49	48	68
12	27	60	57	44	62	36	74	22	34	48	75	47	28	48	129
20	35	60	65	49	82	36	48	23	8	48	107	28	43	18	67
22	9	60	42	95	86	36	111	40	41	48	71	34	42	18	93
29	8	60	49	12	22	24	53	44	32	48	82	30	27	18	70
75	30	12	104	23	9	24	38	46	84	48	57	45	10	18	79
39	63	12	98	31	15	24	45	55	59	48	80	69	22	60	132
20	15	4	114	18	86	6	87	16	54	6	86	17	15	24	81
41	35	4	121	34	55	6	116	26	42	6	76	33	11	24	63
81	69	4	129	48	41	6	120	35	21	6	90	19	35	24	93
70	23	24	133	23	64	36	90	41	29	6	111	27	44	24	77
11	33	36	74	49	64	36	129	16	47	10	99	38	39	24	100
19	56	36	59	49	19	36	142	23	61	10	92	65	49	24	128
23	11	36	71	30	98	24	107	25	27	10	122	27	69	12	51
26	40	36	102	52	78	24	115	22	77	12	48	34	39	12	69
19	47	48	55	59	52	24	111	25	70	12	65	57	76	12	93
21	31	48	43	72	10	24	132	42	57	12	101	67	58	12	132
27	12	48	55	12	50	6	67	11	6	12	68	11	86	10	56
31	49	48	54	15	57	6	56	19	16	12	67	17	65	10	45
11	42	48	23	18	15	6	83	50	9	12	134	11	59	10	31
15	8	48	42	30	40	6	106	35	100	6	86	12	51	10	40
24	33	48	51	17	55	6	86	70	101	6	108	33	29	24	92
44	5	48	97	18	54	24	57	9	46	12	59	43	52	24	127
75	46	48	115	32	70	24	70	30	9	12	87	65	44	24	93
100	17	48	153	58	10	24	105	30	21	12	75	20	52	12	95
115	111	48	144	77	55	24	94	42	46	12	75	30	35	12	116
15	33	36	38	28	99	12	46	49	40	12	109	47	28	12	85
37	46	36	50	29	84	12	72	52	47	12	77	40	66	36	56
54	42	36	100	40	69	12	92	24	58	8	68	62	43	36	81
79	43	48	133	45	60	12	88	46	55	8	75	62	37	36	125
10	46	48	30	46	100	12	70	37	37	8	93	28	39	12	54
11	57	48	34	67	35	12	144	49	40	36	116	39	8	12	98
12	40	48	39	18	32	36	42	56	66	36	58	43	32	12	99
12	52	48	62	27	15	36	60	56	17	36	103	22	44	12	64
12	51	3	58	30	32	36	39	57	13	36	87	40	32	12	70
16	65	3	59	40	5	36	59	59	35	7	129	50	29	12	78
38	18	3	145	12	97	12	33	29	32	7	74	48	18	12	74
10	49	24	39	37	39	12	117	27	22	7	110	12	21	24	39
24	36	24	92	82	44	12	155	27	20	7	108	38	36	24	89
29	22	24	98	41	111	36	92	19	50	6	70	29	87	60	50
60	16	24	156	77	82	36	162	26	46	6	51	42	55	60	62
16	49	6	115	36	94	24	63	56	13	6	88	61	63	60	68
31	54	6	90	54	99	24	83	48	12	6	99	80	76	60	97
15	14	24	63	14	45	6	47	18	35	36	77	35	64	6	63
16	14	24	40	18	6	6	41	27	14	36	79	44	47	6	76
24	10	24	113	19	32	6	69	54	30	36	100	64	34	6	94
26	5	24	60	35	29	8	135	57	30	36	99	16	69	36	65
15	33	5	90	42	65	8	105	15	41	60	42	23	92	36	69
39	15	5	135	36	55	24	103	20	22	60	49	52	93	36	74
40	10	5	120	50	13	24	72	26	34	60	57	17	59	6	57
40	12	5	131	11	36	24	54	51	34	60	60	20	27	6	64
13	53	18	83	20	34	24	71	26	24	12	67	21	21	6	113
31	22	18	88	32	29	24	95	37	7	12	63	20	26	4	74
23	38	6	54	67	11	24	88	15	48	36	62	28	34	4	69
25	28	6	91	20	35	12	82	20	27	36	63	36	62	4	82

Appendix B.

Data set of 116 projects developed by 29 students between the years 2011 and first semester of 2012 used for testing the models. N&C: New and changed lines of code; R: Reused lines of code; PLE: Programming language experience of students (in months); E: Effort (in minutes).

N&C	R	PLE	E	N&C	R	PLE	E	N&C	R	PLE	E	N&C	R	PLE	E
21	39	36	55	20	38	10	62	17	62	8	45	39	46	42	64
39	52	10	58	15	23	24	64	20	89	60	45	59	77	38	66
24	40	60	66	22	20	10	66	34	48	48	47	30	84	19	71
27	28	36	75	43	78	24	66	37	78	48	54	38	102	12	71
32	11	10	76	30	50	12	68	27	44	8	55	50	62	19	74
54	29	36	100	16	43	4	69	24	94	8	57	45	83	18	76
30	47	36	104	29	26	48	73	32	32	60	59	56	21	8	78

N&C	R	PLE	E	N&C	R	PLE	E	N&C	R	PLE	E	N&C	R	PLE	E
46	39	60	120	39	43	8	77	41	80	8	59	23	81	18	79
54	31	60	139	22	27	12	79	39	25	48	67	41	22	12	79
17	42	48	31	17	29	12	80	20	89	60	73	22	96	18	84
16	60	48	32	39	43	8	85	39	20	60	80	43	54	18	85
10	19	36	35	75	7	36	88	39	20	60	90	40	16	19	90
13	43	10	37	28	23	10	91	91	12	8	92	25	63	8	92
27	53	60	43	22	13	12	93	27	44	60	102	30	20	12	95
40	56	48	44	41	22	48	93	20	20	14	105	35	49	36	95
23	13	36	46	65	60	30	93	23	44	14	110	33	48	10	98
11	33	36	48	18	11	12	96	23	13	14	125	20	74	8	100
17	46	48	48	37	12	36	98	45	12	8	143	17	32	12	102
20	37	12	48	50	27	30	99	23	72	54	42	37	17	12	105
15	67	60	53	35	65	10	100	22	99	50	44	42	43	36	105
20	56	24	53	77	19	8	100	18	94	42	45	39	20	36	107
22	65	7	53	27	51	8	101	36	87	50	47	109	30	22	107
27	26	12	53	22	36	12	106	14	75	42	49	14	41	10	110
18	38	12	54	42	43	4	106	33	37	50	51	46	36	11	113
41	96	30	54	32	23	24	107	22	65	11	53	32	15	10	114
19	92	8	57	46	36	7	113	27	104	54	55	65	21	8	114
26	44	48	60	77	51	8	125	28	47	12	59	37	23	36	119
30	45	7	60	40	40	10	145	30	45	11	60	38	20	12	140
26	47	36	61	15	44	48	42	30	60	12	61	46	28	10	173

Cuauhtémoc López-Martín obtained his Ph.D. degree in Computer Science in 2007 at Instituto Politécnico Nacional, México City. Currently he is Researcher Professor at Universidad de Guadalajara, Jalisco, México. His areas of interest are: Software Engineering Education, Software Engineering Processes, Statistics and Machine Learning Techniques applied to Software Engineering and Prediction Models for Software Engineering.

Ivica Kalichanin-Balich obtained his Ph.D. degree in Information Technologies in 2015 at Universidad de Guadalajara, Jalisco, México. His areas of interest are: Statistics and Machine Learning Techniques applied to Software Engineering and Prediction Models for Software Engineering

Rosa Leonor Ulloa-Cazarez is a Ph.D. Student in Information Technologies at Universidad de Guadalajara, Jalisco, México. Her areas of interest are: Online Education, Statistics and Machine Learning Techniques applied to Education and Prediction Models for Education.

Noel García-Díaz obtained his Ph.D. degree in Information Technologies in 2014 at Universidad de Guadalajara, Jalisco, México. He is Researcher Professor at Universidad de Colima as well as at Instituto Tecnológico de Colima, México. His areas of interest are: Statistics and Machine Learning Techniques applied to Software Engineering and Prediction Models for Software Engineering.