Applied Mathematics & Information Sciences
An International Journal

Application of Gamma Classifier to Development Effort Prediction of Software Projects

Cuauhtémoc López-Martín[1], *Itzamá López-Yáñez*[2], *Cornelio Yáñez-Márquez*[3]

[1] Information Systems Department, CUCEA, Guadalajara University, Jalisco, Mexico, 45100.
[2] Telematics Academy, Engineering and Advanced Technologies Interdiscilinary Professional Unit, National Polytechnic Institute, Av. Instituto Politécnico Nacional 2580, Mexico City, Mexico, 07340.
[3] Neural Networks and Unconventional Computing Laboratory, Center for Computing Research, National Polytechnic Institute, Av. Juan de Dios Bátiz s/n, Mexico City, Mexico, 07738.

Received: Feb 8, 2012; Revised Mar. 4, 2012; Accepted Apr. 16, 2012
Published online: 1 Sep. 2012

Abstract: The Gamma Classifier is a novel algorithm, immersed in the Associative Approach to Pattern Recognition, of which the Alpha-Beta BAM is another relevant model. The Gamma Classifier has shown competitive performance in areas such as prediction of atmospheric pollutants, wireless network sensor location, and concrete mix properties forecast. This paper introduces the fist successful application of this model to development effort prediction of software projects. In this sense, an ongoing concern of software managers is to predict how many hours should be spent on a development project, mainly regarding project budgeting and planning. Software managers based typically their predictions on judgment-based techniques; however, models-based techniques (statistical regressions, fuzzy logic, neural networks, or genetic programming) offer a good alternative. In this study, the Gamma Classifier was trained with a data set of 163 software projects and then used for predicting the effort of another data set integrated by 68 projects; all projects were developed by 53 and 21 practitioners respectively. Accuracy result of this classifier was compared with that of a fuzzy logic model and that from a statistical regression model.

Keywords: Development effort prediction, Gamma classifier, associative models, software development.

1. Introduction

The Gamma Classifier is an algorithm of recent proposal, which has shown good performances on such dissimilar areas as the prediction of atmospheric pollutants [1–4], the location of wireless devices in a wireless sensor network, and the forecast of the properties of a concrete mix [5,6]. These promising performances are not the only interesting characteristic of this model. Similarly to other Alpha-Beta Associative Models, the Gamma Classifier follows an unconventional take on the Associative Approach to Pattern Recognition, being able to better cope with difficult problems than other more classically inclined models. Another example of this fitness to difficult problems is the Alpha-Beta Bidirectional Associative Memories (BAMs), which is one of the earlier Alpha-Beta Associative Models and one of the highest performers among them [7–9]. Such unconventional approach and good results have helped the Alpha-Beta BAMs establish the Alpha-Beta Models as a good alternative, helping also to raise awareness of the properties and advantages (as well as limitations) of the Associative Approach to Pattern Recognition.

These similarities in both performance and fitness to different, apparently unrelated areas between the Alpha-Beta BAMs and the Gamma Classifier, served the authors as motivation to try and apply the latter model to a seldom studied problem: that of predicting software development effort in the context of small software development projects. Indeed, the current work presents the first successful application of the Gamma Classifier to software development effort estimation.

The accuracy on software effort prediction (also known as estimation [10]) is a research topic that has attracted much of the interest of the software engineering community during the latest decades (at least from 1966 [11]), and this topic in software engineering can be defined as the

* Corresponding author: e-mail: cuauhtemoc@cucea.udg.mx ilopezy@ipn.mx cyanez@cic.ipn.mx

process of predicting the amount of effort required for the completion of a software artifact. The accuracy of a software effort model is its ability to predict the effort value of a new project closely to the actual one [12].

Based upon if project managers were better educated in prediction techniques, they could improve their effort and schedule prediction capability and credibility [13], this paper contributes with a model for improving the initial effort prediction (where project managers initially should look to improve their chances of project success [13]). Under-estimating a project may lead to under-staffing it, under-scoping the quality assurance effort, and setting short schedule, whereas to overestimating a project, if in a project is given more resources than it really needs without sufficient scope controls it may use them and the project is then likely to cost more than it should, take longer to deliver than necessary, and delay the use of the resources in the next project [14].

The rest of the document is organized as follows. Section 2 is dedicated to exploring the techniques which are currently used to predict software development effort, as well as some measures of performance used to compare the obtained performance. Later, the Gamma Classifier is discussed in section 3, while the experiments done and their results are described and analyzed in section 4. On the other hand, section 5 presents the conclusions and future work, and finally the consulted references are included.

2. Software Development Prediction Techniques

Software development prediction techniques could be classified into two general categories:

1. Expert judgment. This technique implies a lack of analytical argumentation and aims to derive predictions based on the experience of experts on similar projects; this technique is based on a tacit (intuition-based) quantification step [15].
2. Model-based technique. It is based on a deliberate (mechanical) quantification step [15] and it includes models based on statistical regressions (roughly half of all prediction papers tried to build, improve or compare to regression model-based prediction methods [10]), and they include fuzzy logic [16], neural networks [17], genetic programming [18], genetic algorithms [19], and associative memories [20]).

Given that no single prediction technique is best for all situations and that a careful comparison of the results of several approaches is most likely to produce realistic predictions [21–24], this paper compares three models: a classifier, a fuzzy model and a statistical regression model [25, 26]. These models were generated from data of developed projects with practices of Personal Software Process (PSP). This kind of process was selected because it is related to the Capability Maturity Model (CMM) that gives an available description of the goals, methods, and practices needed in international software engineering industrial practice [27], whereas 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 [28]). In addition, the levels of software engineering training could be classified in the small and in the large software projects, and when the approach is related to small projects, the PSP (whose practices and methods are used for delivering quality products on predictable schedules), can be useful. PSP assumes that unless engineers have the capabilities provided by personal training, they cannot properly support their teams or consistently and reliably produce quality products [28].

One of the most usual effort prediction approaches is the construction of a conceptual model involving a set of variables and a set of logical and quantitative relationships between them [12]; in this paper, a classifier named Gamma is trained with a data set of 163 projects developed through years 2005, 2006 and 2007 by 53 practitioners (verification or training stage). This data set was integrated by the following two independent variables: new and changed (N&C) code, and reused code, whereas the dependent variable was the effort. Afterwards, this Gamma classifier was applied to other data set integrated by 68 developed projects through year 2008 by other group integrated by 21 practitioners (validation stage). The experimental design, data of the developed projects, and the names of the 74 practitioners are included in [16].

Accuracy of the Gamma classifier is compared with accuracies of the following two models: a classical statistical regression, which corresponds to the approach most common in parametric prediction models [10], and with a fuzzy logic model. The reasons for comparing results of the classifier against these two models are the following: in accordance to [29], the comparison against a statistical regression model is made because a regression analysis for selecting the significant variables should be done as the default model construction method, on the other hand, data and fuzzy logic model used in this study have already been described in [16].

The hypotheses to be investigated in this paper are the following:

H_1 Effort prediction accuracy of the Gamma Classifier is statistically equal or better than those obtained from a multiple linear regression, and from a fuzzy logic model, when these three models were trained with 163 developed projects inside of a controlled experiment (verification stage).

H_2 Effort prediction accuracy of the Gamma Classifier is statistically equal or better than those obtained from a multiple linear regression, and from a fuzzy logic model, when these three models were applied to a new set of 68 developed projects inside of a controlled experiment (validation stage).

Source lines of code (LOC) remains in favor of many models [30] and researchers commonly use it to correlate

effort. 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 [31]. In this study, the independent variables of the models are 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 project when modifying a previously developed project. The base project is the total LOC of the previous project while the reused code is the LOC of previously developed project that are used without any modification [28].

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 [28]. Hence, it is necessary to always use both the same coding and counting standards. The developed projects for this study followed these two guidelines.

There are many different definitions of software project success [13], one of them is related to this paper: comparing the actual versus predicted elapsed time for the project. Conclusions of an accuracy criteria analysis recommends the use of the Magnitude of Error Relative to the estimate (MER) to evaluate and compare prediction models [32], which is defined as follows:

$$MER_i = \frac{|AE_i - PE_i|}{PE_i} \tag{1}$$

where AE stands for Actual Effort and PE means Predicted Effort.

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

$$MMER = \frac{1}{n}\sum_{i=1}^{n} MER_i \tag{2}$$

A study of the most referenced journal with respect to effort prediction work [10] showed that the diversity of prediction approaches is very high and increasing, and this increase is illustrated by the rising proportion of approaches related to techniques based upon models. A large part of the research efforts involves the development of statistical models based on historical data [33].

Experimental results related to the application of pattern recognition techniques have shown the effectiveness for the particular application of software effort prediction [34]. Other classifiers have been applied for predicting software defects [35], whereas the Gamma Classifier has been used for predicting air quality, the location of wireless devices in a wireless sensor network, and the properties of a concrete mix [1–6].

3. Gamma Classifier

The basis of the Gamma classifier is the gamma operator, hence its name. The gamma operator is based on the alpha, beta, and u_β operators and their properties, in particular when dealing with binary patterns coded with the modified Johnson-Möbius code.

The alpha and beta operators are defined in tabular form, taking into account the definitions of the sets A and B, as is shown in Table 1. Thus, let there be the sets $A = \{0, 1\}$ and $B = \{0, 1, 2\}$.

$\alpha : A \times A \to B$		
x	y	$\alpha(x, y)$
0	0	1
0	1	0
1	0	2
1	1	1

$\beta : B \times A \to A$		
x	y	$\beta(x, y)$
0	0	0
0	1	0
1	0	0
1	1	1
2	0	1
2	1	1

Table 1 Definition of the Alpha and Beta operators

The modified Johnson-Möbius code allows to code a set of real numbers into binary representations by, first, subtracting the lesser negative (if there are negatives) from all numbers, leaving only non-negative reals; then the numbers are scaled up in order to leave only non-negative integers (truncating the remaining decimals if necessary); and then $e_m - e_j$ zeros are concatenated with e_j ones, where e_m is the greatest non-negative integer number to be coded, and e_j is the current non-negative integer number to be coded.

For instance, let $D \subset \mathbb{R}$ be defined as $D = \{1.6, 1.9, 0.3, 0.8, -0.1\}$; now, lets use the modified Johnson-Möbius code (as explained above) to convert the elements of D into binary vectors.

1.Subtract the minimum:

$$D \longrightarrow T \subset \mathbb{R},$$
$$T = 1.7, 2.0, 0.4, 0.9, 0.0 \tag{3}$$

Since -0.1 is the minimum number in the set, the members of the transformed set T are obtained by subtracting -0.1 to each member of D (which is the same as adding 0.1): $t_i = d_i - (-0.1) = d_i + 0.1$. Notice that this step is particularly useful for handling negative numbers, since now there will be only non-negative numbers, with the minimum being 0.

2.Scale up the numbers:

$$T \longrightarrow E \subset \mathbb{Z}^+,$$
$$T = 17, 20, 4, 9, 0 \tag{4}$$

Since there is only one decimal digit, it is enough to multiply each number by 10 to obtain integers, thus the members of the integers set E are calculated as: $e_i = t_i \times 10$.

3. Concatenate $e_m - e_j$ zeros with e_j ones, where e_m is the maximum non-negative integer number to be coded, and e_j is the current non-negative integer number to be coded:

$$E \longrightarrow C = \left\{c_i | c_i \in A^{20}\right\},$$
$$C = \left\{\begin{matrix}[00011111111111111111] \\ [11111111111111111111] \\ [00000000000000001111] \\ [00000000000111111111] \\ [00000000000000000000]\end{matrix}\right\} \quad (5)$$

Given that the maximum number in the set E of non-negative integers is 20, all binary vectors have 20 components, i.e. they all are 20 bits long. In other words, each member of the C set is a binary vector which belongs to the set resulting from applying 20 times the cross product of the set $A = \{0, 1\}$ to itself.

For example, $e_1 = 17$ is converted into its binary representation c_1 by appending $e_m - e_1 = 20 - 17 = 3$ zeroes "000", followed by $e_1 = 17$ ones "11111111111111111", which gives the final vector: "00011111111111111111".

This is because the maximum number in set E is 20, thus $e_m = 20$; and the number to be converted is $e_1 = 17$.

The unary u_β operator receives as input an n-dimensional binary vector $\mathbf{x}$, outputs a non-negative integer number, and is calculated as shown below:

$$u_\beta = \sum_{i=1}^{n} \beta(x_i, x_i) \quad (6)$$

Thus, if $\mathbf{x} = [100110]$ then:

$$\begin{aligned} u_\beta(\mathbf{x}) &= \beta(1,1) + \beta(0,0) + \beta(0,0) \\ &\quad + \beta(1,1) + \beta(1,1) + \beta(0,0) \\ &= 1 + 0 + 0 + 1 + 1 + 0 = 3 \end{aligned} \quad (7)$$

The generalized gamma operator γ_g which takes as input two binary patterns $\mathbf{x} \in A^n$ and $\mathbf{y} \in A^m$, with $n, m \in \mathbb{Z}^+$, $n \leq m$, and a non-negative integer number θ; and gives a binary number as output, can be calculated as follows:

$$\gamma_g(\mathbf{x}, \mathbf{y}, \theta) = \begin{cases} 1 & \text{if } m - u_\beta\left[\alpha(\mathbf{x}, \mathbf{y}) \mod 2\right] \leq \theta \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

where $\mod 2$ indicates the usual modulo 2 operator.

To better illustrate how the generalized gamma operator works, let us step through an example of its application. Thus, let us find $\gamma_g(\mathbf{x}, \mathbf{y}, \theta)$ if $\mathbf{x} = [11100]$, $\mathbf{y} = [10110]$, and $\theta = 2$.

First, we have that $\alpha(\mathbf{x}, \mathbf{y})$ = [1 2 1 0 1]; then [1 2 1 0 1] mod 2 = [1 0 1 0 1]; now

$$\begin{aligned} u_\beta[10101] &= \beta(1,1) + \beta(0,0) + \beta(1,1), \\ &\quad + \beta(0,0) + \beta(1,1) \\ &= 1 + 0 + 1 + 0 + 1 = 3 \end{aligned} \quad (9)$$

and since $m = 5$, $5 - 3 = 2$; given that $2 \leq \theta = 2$, the result is 1.

The gamma classifier algorithm is depicted as follows:

Let $k, m, n, p \in \mathbb{Z}^+$; $\{\mathbf{x}^\mu | \mu = 1, 2, \ldots, p\}$ be the fundamental (learning) pattern set with cardinality p, where $\forall \mu \mathbf{x}^\mu \in \mathbb{R}^n$, and let $\mathbf{y} \in \mathbb{R}^n$ be an n-dimensional real-valued pattern to be classified. It is assumed that the fundamental set is partitioned into m different and mutually exclusive classes, each class having a cardinality k_i, $i = 1, 2, \ldots, m$, and thus $\sum k_i = p$. In order to classify $\mathbf{y}$, the following steps are implemented:

1. Code the fundamental set with the modified Johnson-Möbius code, obtaining the following value of e_m for each component:

$$e_m(j) = \bigvee_{i=1}^{p} x_j^i \quad (10)$$

2. Compute the following stop parameter:

$$\rho = \bigwedge_{j=1}^{n} e_m(j) \quad (11)$$

3. Code y with the modified Johnson-Möbius code, using the same parameters used with the fundamental set (step 1). If any y_j is greater than the corresponding $e_m(j)$, the γ_g operator will use such y_j instead of m (according to equation 8).
4. Transform the index of all fundamental patterns into two indices, one for the class they belong to, and another for their position in the class (i.e. $\mathbf{x}^\mu$ which belongs to class i becomes $\mathbf{x}^{i\omega}$).
5. Initialize θ to zero.
6. Do $\gamma_g(x_j^{i\omega}, y_j, \theta)$ for each component of the fundamental patterns in each class.
7. Compute the weighted sum c_i for each class, as follows:

$$c_i = \frac{\sum_{\omega=1}^{k_i} \sum_{j=1}^{n} \gamma_g(x_j^{i\omega}, y_j, \theta)}{k_i} \quad (12)$$

8. If there is more than one maximum among the different c_i, increment θ by 1 and repeat steps 6 and 7 until there is a unique maximum, or the stop condition $\theta \geq \rho$ is fulfilled.

9.If there is a unique maximum, assign $\mathbf{y}$ to the class corresponding to such maximum:

$$C_y = C_j \text{ such that } \bigvee_{i=1}^{m} c_i = c_j \qquad (13)$$

10.Otherwise, assign $\mathbf{y}$ to the class of the first maximum.

In order to train the Gamma classifier with the data, some preprocessing is needed. First, a 2-dimensional vector is built with both independent variables, new and changed code and reused code. The class associated to each pattern is the corresponding value of the dependent variable: the effort.

4. Experiments

The experimental results obtained by training and testing the three models (statistical regression model, fuzzy model, and the Gamma Classifier) are shown in Table 2, for both verification and validation stages.

Stage	Statistical Regression Model	Fuzzy Model	Gamma Classifier
Verification	0.27	0.25	0.22
Validation	0.28	0.28	0.28

Table 2 Experimental results by model, in MMER

The ANOVA for MER from the 163 projects showed that there was a statistically significant difference amongst the accuracy of prediction for the three models (p-value of Table 3 is less than 0.05). A means plot is useful for showing which means are significantly different from which others; in Figure 1 can be observed that the Gamma Classifier (GC) had the best accuracy (lower MMER), which indicates that this classifier can be used for predicting the effort.

Source	Sum of squares	Degrees of freedom	Mean square	F-ratio	p-value
Between groups	0.508	2	0.2843	8.30	0.0003
Within groups	14.857	485	0.0306		
Total	15.366	487			

Table 3 ANOVA Table for MER by Model (verification stage)

However, the validity of MER ANOVA is based on the analysis of the following three assumptions of residuals:

1.Independent samples: The group of developers was made up of separate practitioners and each of them developed its own projects, hence the data are independent.

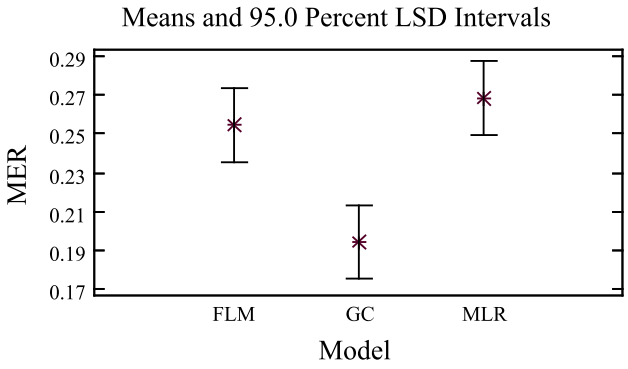

Figure 1 Means plot of MER ANOVA (verification stage)

2.Equal standard deviations: In a plot of this kind the residuals should fall roughly in a horizontal band centered and symmetric about the horizontal axis (as shown in Figure 2), and
3.Normal populations: A normal probability plot of the residuals should be roughly linear (as shown in Figure 3).

Hence, the three assumptions for residuals in the actual data set were considered as met.

In an additional test, since a p-value = 0.0001 of the Kruskal-Wallis test resulted less than 0.05, there was a statistically significant difference between the medians of the models at the 95% confidence level.

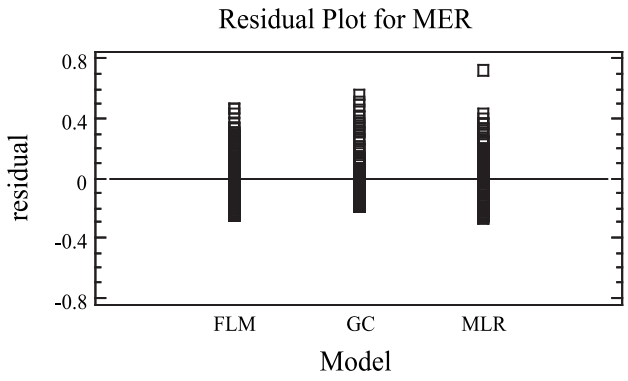

Figure 2 Equal standard deviation plot of MER ANOVA (verification stage)

A second group integrated by 21 practitioners developed 68 projects that were used for validating the three models. Once the three models for predicting the effort were applied to these data, the MER by project as well as the MMER by model were calculated.

In accordance with the ANOVA for MMER models that is depicted in Table 4, there was not a statistically

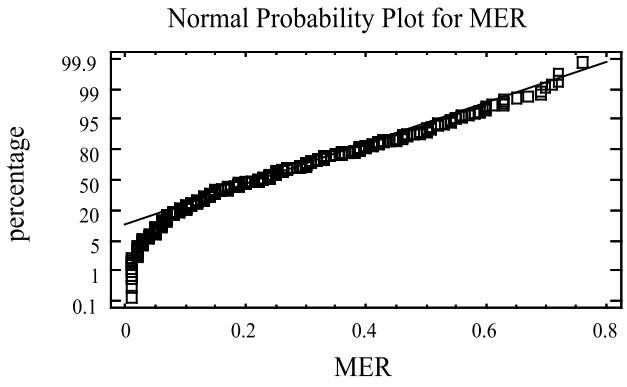

Figure 3 Normality plot of MER ANOVA (verification stage)

significant difference amongst the accuracy of prediction for the three models (p-value is greater to 0.05) at 95% of confidence.

Source	Sum of squares	Degrees of freedom	Mean square	F-ratio	p-value
Between groups	0.0471	2	0.0236	0.61	0.5426
Within groups	7.7529	201	0.0385		
Total	7.8003	203			

Table 4 ANOVA Table for MER by Model (validation stage)

ANOVA plots for equal standard deviation as well as for normality were met (standard deviations were similar and that their data showed normality).

As additional test, since a p-value = 0.8695 of the Kruskal-Wallis test resulted greater than 0.05, there was not a statistically significant difference between the medians of the models at the 95% confidence level.

5. Conclusions and Future Research

Given 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 predictions [22–24], three models were applied to the problem of software development effort estimation in this work, and their respective performances were compared. One of such models was applied for the first time to this particular problem: the Gamma Classifier.

Two samples of 163 and 68 projects were used for verifying and validating respectively the models; these projects were developed by 53 and 21 practitioners respectively. The 231 projects were developed following practices of the Personal Software Process which allows the instrumentation of the Capability Maturity Model (CMM) at a personal level. The independent variables of the models were new and changed code as well as reused code, while the dependent variable was the effort.

The accepted hypotheses in this research were the following:

H_1 Effort prediction accuracy of the Gamma Classifier is better than those obtained from a multiple linear regression, and from a fuzzy logic model, when these three models were trained with 163 developed projects inside of a controlled experiment.

H_2 Effort prediction accuracy of the Gamma Classifier is statistically equal than those obtained from a multiple linear regression, and from a fuzzy logic model, when these three models were applied to a new set of 68 developed projects inside of a controlled experiment.

Based on these accepted hypotheses, the Gamma Classifier could be successfully used for predicting the effort of software projects when both new and changed code, and reused code are used as independent variables.

Given this competitive performance offered by the Gamma Classifier, several extensions to the current work present themselves for future research. On one hand, the performance of the Gamma Classifier on this problem can be further explored by trying different independent variables, on the same set of software development projects, as well as applying this model to other data sets. On the other hand, there are several associative models which appear to have similar performance to the Gamma Classifier. One of these models is that of the Alpha-Beta Bidirectional Associative Memories (BAMs), which could also be applied to the current data set, with the current combination of independent and dependent variables, as well as with different combinations of variables. Additionally, the Alpha-Beta BAMs could be applied to other related data sets.

Acknowledgement

The authors would like to thank the CUCEA of Guadalajara University, Programa de Mejoramiento del Profesorado (PROMEP), Instituto Politécnico Nacional (Secretaría Académica, COFAA, SIP, UPIITA, and CIC), the CONACyT, SNI, and the ICyTDF (grants PIUTE10-77 and PICSO10-85) for their economical support to develop this work.

References

[1] I. López-Yáñez, A.J. Argüelles-Cruz, O. Camacho-Nieto, C. Yáñez-Márquez, Pollutants Time-Series Prediction Using the Gamma Classifier. International Journal of Computational Intelligence Systems, Atlantis Press, Vol. **4**, Issue 4, 2011, pages 680-711. ISSN: 1875-6883

[2] I. López-Yáñez, C. Yáñez-Márquez, V.M. Silva-García, Forecasting Air Quality Data with the Gamma Classifier, Pattern Recognition. Peng-Yeng Yin (Ed.), INTECH, 2009. ISBN: 978-953-307-014-8. Available from: http://sciyo.com/articles/show/title/forecasting-air-quality-data-with-the-gamma-classifier

[3] I. López-Yáñez, C. Yáñez-Márquez, O. Camacho-Nieto, A.J. Argüelles-Cruz, Prediction of Air Contaminant Concentration Based on an Associative Pattern Classifier. Revista Facultad de Ingeniera - Universidad de Antioquia, No. **60**, 2011, pages 32-41. ISSN: 0120-6230

[4] C. Yáñez-Márquez, I. López-Yáñez, G. de la L. Sáenz-Morales, Analysis and Prediction of Air Quality Data with the Gamma Classifier. Lecture Notes in Computer Science, Springer Verlag, Vol. **5197**, 2008, pages 1611-3349. DOI: 10.1007/978-3-540-85920-8_79

[5] I. López-Yáñez, Teoría y aplicaciones del Clasificador Asociativo Gamma (In Spanish). PhD Thesis, Center for Computing Research, National Polytechnics Institute, 2011, México.

[6] I. López-Yáñez, Clasificador Automático de Alto Desempeño (In Spanish). MSc Thesis, Center for Computing Research, National Polytechnics Institute, 2007, México.

[7] M.E. Acevedo, C. Yáñez-Márquez, M.A. Acevedo, Associative Models for Storing and Retrieving Concept Lattices. EURASIP Mathematical Problems in Engineering, Hindawi, Volume 2010, 2010, Article ID 356029. ISSN (printed): 1024-123X. ISSN (electronic): 1563-5147. DOI: 10.1155/2010/356029. Available from: http://www.hindawi.com/journals/mpe/2010/356029.html

[8] M.E. Acevedo-Mosqueda, C. Yáñez-Márquez, I. López-Yáñez, Alpha-Beta Bidirectional Associative Memories: Theory and Applications. Neural Processing Letters, Springer-Verlag Berlin Heidelberg, Vol. **26**, No. 1, 2007, pages 1-40. ISSN: 1370-4621. DOI:10.1007/s11063-007-9040-2

[9] M.E. Acevedo-Mosqueda, C. Yáñez-Márquez, I. López-Yáñez, Complexity of Alpha-Beta Bidirectional Associative Memories. Lecture Notes in Computer Science, Springer-Verlag Berlin Heidelberg, LNCS **4293**, pages 357-366. ISSN: 0302-9743. DOI:10.1007/11925231_34

[10] M. Jørgensen, M. Shepperd, A Systematic Review of Software Development Cost Estimation Studies. IEEE Transactions on Software Engineering, Vol. **33**, 2007, pages 33-53. DOI: 10.1109/TSE.2007.256943

[11] E.A. Nelson, Management Handbook for the Estimation of Computer Programming Costs. AD-A648750, Systems Development Corp. 1966.

[12] S. Bibi, I. Stamelos, L. Angelis, Combining probabilistic models for explanatory productivity estimation. Journal of Information and Software Technology, Elsevier, Vol. **50**, 2008, pages 656-669. DOI: 10.1016/j.infsof.2007.06.004

[13] J.M. Verner, W.M. Evanco, N. Cerpa, State of the practice: An exploratory analysis of schedule estimation and software project success prediction. Journal of Information and Software Technology, Elsevier, Vol. **49**, 2007, pages 181-183. DOI:10.1016/j.infsof.2006.05.001

[14] T.I.F. De Barcelos, J.D. Simies da Silva, N. Sant Anna, An investigation of artificial neural networks based prediction systems in software project management. Journal of Systems and Software, Elsevier, Vol. **81**, 2008, pages 356-367. DOI: 10.1016/j.jss.2007.05.011

[15] M. Jørgensen, Forecasting of Software Development Work Effort: Evidence on Expert Judgment and Formal Models. Journal of Forecasting, Elsevier, Volume **23**, Issue 3, 2007, pages 449-462. DOI: 10.1016/j.ijforecast.2007.05.008

[16] C. Lopez-Martin, A fuzzy logic model for predicting the development effort of short scale programs based upon two independent variables. Journal of Applied Soft Computing, Elsevier, Vol. **11**, Issue 1, 2011, pages 724-732. DOI: 10.1016/j.asoc.2009.12.034

[17] C. López-Martín, Applying a general regression neural network for predicting development effort of short-scale programs. Journal of Neural Computing and Applications, Springer, Vol. **20**, No. 3, 2011, pages 389-401. DOI: 10.1007/s00521-010-0405-5

[18] W. Afzal, R. Torkara, On the application of genetic programming for software engineering predictive modeling: A systematic review. Journal of Expert Systems with Applications, Elsevier, Vol. **38**, Issue 9, 2011, pages 11984-11997. DOI:10.1016/j.eswa.2011.03.041

[19] Sun-Jen, H., Nan-Hsing, Ch., Li-Wei, Ch. 2008. Integration of the grey relational analysis with genetic algorithm for software effort estimation. Journal of Operational Research, Elsevier, Vol. **18**, pages 898-909. DOI: 10.1016/j.ejor.2007.07.002

[20] Y. Kultur, B. Turhan, A. Bener, Ensemble of neural networks with associative memory (ENNA) for estimating software development costs. Journal of Knowledge-Based Systems, Elsevier, Vol. **22**, 2009, pages 395-402. DOI: 10.1016/j.knosys.2009.05.001

[21] B. Boehm, Ch. Abts, S. Chulani, Software development cost estimation approaches: A survey. Journal of Annals of Software Engineering, Springer Verlag, Vol. **10**, 2000, pages 177-205. DOI: 10.1023/A:1018991717352

[22] D.H. Wolpert, W.G. Macready, No free lunch theorems for search. IEEE Transactions on Evolutionary Computation, 1997.

[23] D.H. Wolpert, et al., No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation, 1997.

[24] M. Sewell, No Free Lunch Theorems. 2006. Available at: http://www.no-free-lunch.org/

[25] Z. Xu, T.M. Khoshgoftaar, Identification of fuzzy models of software cost estimation. Journal of Fuzzy Sets and Systems, Elsevier, Vol. **145**, 2004, pages 141-163. DOI: doi:10.1016/j.fss.2003.10.008

[26] C. López-Martín, C. Yáñez-Márquez, A. Gutiérrez-Tornés, Predictive Accuracy Comparison of Fuzzy Models for Software Development Effort of Small Programs. Journal of Systems and Software, Elsevier, Vol. **81**, issue 6, 2008, pages 949-960. DOI: 10.1016/j.jss.2007.08.027.

[27] M. Manish-Agrawal, K. Chari, Software Effort, Quality, and Cycle Time: A Study of CMM Level 5 Projects. IEEE Transactions on Software Engineering, Vol. **33**, 2007, pages 145-156. DOI: 10.1109/TSE.2007.29

[28] W. Humphrey, A Discipline for Software Engineering, Addison Wesley. 1995.

[29] B. A. Kitchenham, E. Mendes, G.H. Travassos, Cross versus Within-Company Cost Estimation Studies: A Systematic Review. IEEE Transactions Software Engineering, Vol. **33**, No. 5, 2007, pages 316-329. DOI: 10.1109/TSE.2007.1001

[30] S. D. Sheetz, D. Henderson, L. Wallace, Understanding developer and manager perceptions of function points and source lines of code. The Journal of Systems and Software, Elsevier, Vol. **82**, 2009, pages 1540-1549. DOI:10.1016/j.jss.2009.04.038

[31] R.E. Park, Software Size Measurement: A Framework for Counting Source Statements. Software Engineering Institute, Carnegie Mellon University. 1992, CMU/SEI-92-TR-020

[32] T. Foss, E. Stensrud, B. Kitchenham, L. Myrtveit, A Simulation Study of the Model Evaluation Criterion MMRE. IEEE Transactions on Software Engineering. Vol. **29**, 2003, pages 985-995. DOI: 10.1109/TSE.2003.1245300

[33] N., Mittas, L. Angelis. Comparing cost prediction models by resampling techniques. The Journal of Systems and Software, Elsevier, Vol. **81**, 2008, pages 616-632. DOI: 10.1016/j.jss.2007.07.039

[34] L.C. Briand, V.R. Basili, W.M. Thomas, A Pattern Recognition Approach for Software Engineering Data Analysis. IEEE Transactions on Software Engineering, Vol. **18**, 1992, pages 931-942. DOI: 10.1109/32.177363

[35] S. Bibi, G. Tsoumakas, I. Stamelos, I. Vlahavas, Regression via Classification applied on software defect estimation. Journal of Expert Systems and Applications, Elsevier, Vol. **34**, 2008, pages 2091-2101. DOI: 10.1016/j.eswa.2007.02.012

Cuauhtémoc López-Martín is a Professor-Researcher at the Information Systems Department, CUCEA Guadalajara University. He received his Bachelor degree in Computer Engineering at the Universidad Autónoma de Tlaxcala (1995), his MSc degree in Information Systems at the University of Guadalajara (2000), and his PhD degree in Computer Science at the National Polytechnics Institute (IPN) Center for Computing Research (CIC) in 2007, with mention of Honor.

Areas of interest: Systems Modeling and Simulation, Software Engineering, in particular Software Quality, Software Development Effort Estimation, and Personal software process statistical control.

Itzamá López-Yáñez was born in Obregon City, Sonora, Mexico. He received his Bachelor degree as Information Systems Engineer (2003) at Monterrey Institute of Technology and Superior Studies (ITESM), Campus Sonora Norte. He obtained the MSc (2007) and PhD (2011) degrees on Computer Sciences at National Polytechnics Institute (IPN) Center for Computing Research (CIC), both with mention of Honor. Currently he is both a Professor at, and the President of the Telematics Academy at IPN Engineering and Advanced Technologies Interdiscilinary Professional Unit (UPIITA).

Areas of interest: Associative Memories, Neural Networks, Software Engineering, and Pattern Classification, in particular the Gamma classifier.

Cornelio Yáñez-Márquez is of Mexican nationality; he obtained his Bachelor degree (1989) on Physics and Mathematics at National Polytechnics Institute (IPN) Physics and Mathematics Superior School. His MSc (1995) and PhD (2002) degrees were received at IPN Center for Computing Research (CIC). Currently a Researcher Professor, Titular C, at IPN CIC. He was granted the Lázaro Cárdenas Award by the President of the Republic. Member of the National Researchers System (SNI).

Areas of interest: Associative Memories, Neural Networks, Mathematical Morphology, and Software Engineering.