Proceedings of International Joint Conference on Neural Networks, Orlando, Florida, USA, August 12-17, 2007

Comparison of Artificial Neural Network and Regression Models in Software Effort Estimation

Iris Fabiana de Barcelos Tronto, José Demísio Simões da Silva, Nilson Sant'Anna

***Abstract*— Good practices in software project management are basic requirements for companies to stay in the market, because the effective project management leads to improvements in product quality and cost reduction. Fundamental measurements are the prediction of size, effort, resources, cost and time spent in the software development process. In this paper, predictive Artificial Neural Network (ANN) and Regression based models are investigated, aiming at establishing simple estimation methods alternatives. The results presented in this paper compare the performance of both methods and show that artificial neural networks are effective in effort estimation.**

I. Introduction

The continuous hardware and software development, jointly with the world economical interaction has contributed to the competitiveness increase between producing and delivering companies of software product and services. In addition, there has been a growing need to produce low cost high quality software in a short time.

A quality level and international productivity can be achieved through the use of effective software management process, focalizing people, product, process, and project. The project requires planning and accompaniment supported by a group of activities, among which the estimates are fundamental, because they supply a guide for the other activities. The predictive process involves the set of procedures presented in the Figure 1 [1]. Software size estimates are important to determine the software project effort [2], [3,] [4], [5]. However, according to the last research reported by the Brazilian Ministry of Science and Technology - MCT, in 2001, only 29% of the companies accomplished size estimates and 45,7% accomplished software effort estimate [6]. There is not a specific study that identifies the causes of the effort low estimates index, but the reliability level of the models can be a possible cause. These data presented by MCT evidences the importance to use an effort estimate alternative approach, through which one can have reliable estimates with simple execution model.

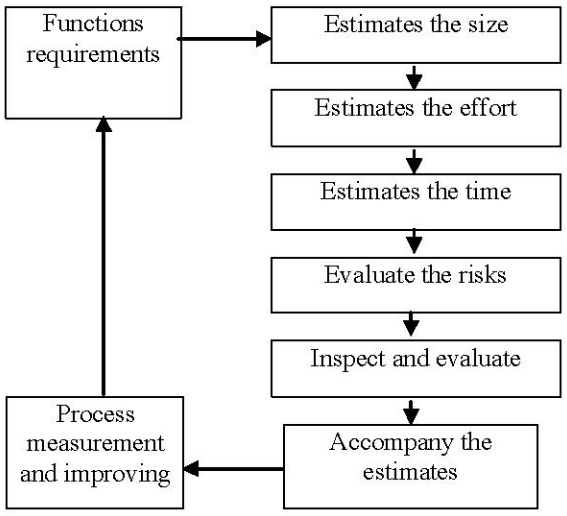

Fig. 1. The predictive process [1]

Predicting software development effort with high precision is still a great challenge for project managers. Consequently, there is an ongoing, high level activity in this research field in order to build, to evaluate, and to recommend prediction techniques [7], [8], [9], [10], [11], [12] [13], [14]. A large number of different predictive models (estimation models and predictive models are considered synonyms) have been proposed over the last years. They range from mathematical functions (e.g. regression analysis [11] and COCOMO [7]) to machine learning models - ML (e.g., estimation by analogy [9], clustering techniques [10], Artificial Neural Network – ANN [14], and regression trees [15]). In contrast to a regression model which is defined by a mathematical formula, ML models are not defined by a mathematical formula but may take on many different shapes.

Despite the number of research activities, there is still a doubt to advise practitioners as to what prediction models they should select, because the studies have not converged to similar answers.

There are a number of factors that should be considered in the selection of a prediction technique, and it is likely that trade-offs will need to be made in the process. Technique selection should be driven by both organizational needs and capability. In terms of need, the most common aim is to

Manuscript sent January 30, 2007. This work was supported in part by the Brazilian National Institute for Space Research - INPE. The authors would like to thanks the CAPES for the financial support.

I. F. Barcelos Tronto is with the Brazilian National Institute for Space Research, Sao José dos Campos, BRA (phone: +551138138684; e-mail: iris_barcelos@ lac.inpe.br).

J.D. Simões da Silva is with the Brazilian National Institute for Space Research, Sao José dos Campos, BRA (phone: +551239456543; e-mail: demisio@ lac.inpe.br)..

N. Sant'Anna is with the Brazilian National Institute for Space Research, Sao José dos Campos, BRA (phone: +551239456537; e-mail: nilson@ lac.inpe.br).

maximize the accuracy in prediction; however, other issues may also need to be considered. For instance, a technique that produces slightly less accurate but generally more robust models might be preferred, especially in cases where the organizations do not have access to locally calibrated, well-behaved data sets. While it is very positive that more sophisticated (and potentially more useful) techniques are being employed to build predictive models, genuine benefits will be achieved if the techniques are appropriately used.

The artificial neural network approach is adaptable and nonparametric; thus predictive models can be tailored to the data at a particular site. Once the ANN is not limited to a linear function, it can deal more successfully with observations that lie far from the best straight line. In this paper, however, the main focus is on investigating the accuracy of the predictions using ANN-based and regression models. A case study was performed to examine the potential of two approaches: a multi-layer perceptron neural network and a linear regression model, using the COCOMO database (7).

The paper is organized as follows: Section 2 provides some background information on the different prediction techniques that are used as the basis for the study accomplished. It is followed by a description of the ANN and regression techniques (Sections 3 and 4, respectively) and the case study itself presented in the Section 5. The paper concludes with discussion on the significance of the results and ideas to the continuity of the research.

II. THE PREDICTION TECHNIQUES

Accurate and consistent prediction of resource requirements is a crucial component in the effective management of software projects. Despite of extensive research over the last 20 years, the software community is still significantly challenged when it comes to effective resource prediction. On the whole, research efforts have focused on the development of techniques that are quantitatively based, in an effort to remove or reduce subjectivity in the estimation process. Examples of this work include the original parametric and regression-based models: Function Points Analysis [16], COCOMO Models [7], [17] and Ordinal Regression Model [11].

However, other techniques for the exploratory data analysis, such as clustering, case-based reasoning and ANN have been effective as a means of predicting software project effort. Zhong et al. [10] describe the use of clustering to predict software quality. A case-based approach called ESTOR was developed for software effort estimation [18]. Vicinanza et al. have shown that ESTOR was comparable to a specialist and it performs significantly better than COCOMO and Function Points on restricted samples of problems. Karunanithi et al. [19] reports the use of neural networks for predicting software reliability, including experiments with both feedforward and Jordan networks with a cascade correlation learning algorithm. Wittig and Finnie [20] describe their use of back propagation learning algorithms on a multilayer perceptron in order to predict development effort. An overall error rate (MMRE) obtained which compares favorably with other methods.

Another study by Samson et al. [14] uses an Albus multiplayer perceptron in order to predict software effort. They use Boehm's COCOMO dataset. The work compares linear regression with a neural networks approach using the COCOMO dataset. But, both approaches seem to perform badly with MMRE of 520,7% and 428,1%, respectively.

Srinivasan and Fisher [14] also report the use of a neural network with a back propagation learning algorithm. They found that the neural network outperformed other techniques and gave results with MMRE= 70%. However, it is not clear how the dataset was divided for training and validation purposes.

Khoshgoftaar et al. [21] presented a case study considering real time software to predict the testability of each module from source code static measures. They consider ANNs as promising techniques to build predictive models, because they are capable of modeling non linear relationships.

Finally, in the last years, a great interest on the use of ANNs has grown. ANNs have been successfully applied to several problem domains, in areas such as medicine, engineering, geology, and physics, in general to design solutions for estimate problems, classification, control, etc. They can be used as predictive models because they are modeling techniques capable of modeling complex functions.

In this work, the artificial neural networks methodology is used to predicting software development effort (in man-hour) from the project size (given by the amount of source code lines). A comparative analysis was accomplished between a regression model and an ANN model that were calibrated and tested in this study.

III. ARTICIAL NEURAL NETWORKS

ANNs are massively parallel systems inspired by the architecture of biological neural networks, comprising simple interconnected units (artificial neurons). The neuron computes a weighted sum of its inputs and generates an output if the sum exceeds a certain threshold. This output then becomes an excitatory (positive) or inhibitory (negative) input to other neurons in the network. The process continues until one or more outputs are generated. An artificial neuron computes the weighted sum of its n inputs, and generates an output of y.

The neural network results from the arrangement of such units in layers, which are interconnected one to another. The resulting architectures solve problems by learning the characteristics of the available data of related to the problem. There exist many different learning algorithms. Feed-forward Multilayer Perceptrons are the most commonly used form of ANN, although many more sophisticated neural networks have been proposed. Multi-layer architectures are mostly trained by the error back propagation algorithm that requires a differentiable activation function.

The ANN is initialized with random weights and gradually learns the relationships implicit in a training data set by adjusting its weights when presented to these data. Among the several available training algorithms the error back propagation is the most used by software metrics researchers.

In general the studies concerned with the use of ANNs to predict software development effort have focused mostly on the accuracy comparison of algorithmic models rather than on the suitability of the approach for building software effort prediction systems. An example is the work of Witting and Finnie [22]. They explore the use of a multilayer neural network on the Desharnais and Australian Software Metrics Association (ASMA) data sets. For the Desharnais data set they randomly split the projects three times between 10 test and 71 training (a procedure we largely follow in our analysis). The results from three validations sets are aggregated and yield a high level of accuracy (Desharnais MMRE = 27% and ASMA MMRE = 17%) although some outlier values are excluded. However, other factors such as exploratory value and configurability are equally important and also need to be investigated.

IV. LINEAR REGRESSION

Linear regression attempts at finding linear relationship between one or more predictor parameters and a dependent variable, minimizing the mean square of the error across the range of observations in the data set. Some researchers have tried building simple local models, e.g. Kok et al. [23], using this type of approach. The philosophy is essentially one of solving local prediction problems before attempting at constructing universal models. The resulting prediction systems take the form:

$$Y_{ext} = \beta_0 + \beta_1 X_1, \ldots, \beta_n X_n \qquad (1)$$

where Y_{est} is the estimated value and $X_1, \ldots, X_n$ are independent variables, for example project size (in source code lines), that the estimator has found to significantly contribute to the prediction of effort. A disadvantage with this technique is its vulnerability to extreme outlier values although robust regression techniques, that are less sensitivity to such problems, have been successfully used [5]. Another potential problem is the impact of co-linearity – the tendency of independent variables to be strongly correlated with one another – upon the stability of a regression type prediction system.

V. THE CASE STUDY

The analysis undertaken in this study deals with a set of measures taken from COCOMO dataset [7]. The aim of the case study was to compare two different prediction techniques: ANN and regression models.

In this section, we describe the data set used in our analysis, summarize the data preparation activities, and explain the approach followed in to build the models and application, and discusses the results.

A. The dataset

The dataset used in this work is COCOMO a public available data set consisting of a total of 63 projects at the time of this study. It was used for describing and testing one of the most important effort estimative methods: the COCOMO model, implemented by Boehm [7]. Furthermore, various methods have been already applied on it [11]. The variables that describe each project are presented in [7]. The effort is represented by the variable EsforcoIT (the amount of man-hour for the software integration and test phase). The systems are mainly written using the programming languages COBOL, PLI, HMI and FORTRAN. The area types are mainly business, scientific and system software.

B. The preparation of the independent variables

Number All of the 63 completed projects were used in our analysis. The dataset doesn't include effort measures for development phase, but the total effort is given by the variable EsforçoIT (which is MMACT on COCOMO dataset). However, the objective is to generate a model that allows predicting effort for each development phase. Thus, we calculated the effort for the requirements specification, product design, detailed design, code and unit test, and integration and test phases, based on the MMACT and in the effort indicators given in [7]. The EsforçoIT was considered to be the dependent variable.

The choice of the independent variables was accomplished using the General Regression Models -GRM module, implemented by the software package STATISTICA. It was implemented the best-subset model-building technique for finding the "best" model from a number of possible models. The subset adjusted R-square statistic allowed direct comparisons and choice of the “best" subset between ten models. The independent variables that compose this model are RELY, ACAP, AEXP, MODP, and TOTKDSI (software size in source lines of code). The effort driver variables considered in this work and a briefly description are presented in Appendix A.

We performed a stepwise regression for the COCOMO projects using the variables presented above. The stepwise regression builds a prediction model by adding to the model, at each stage, the variable with the highest partial correlation to the response variable, taking into account all variables currently in the model. Its aim is to find the set of predictors that maximize F. F assesses whether the regressors, taken together, are significantly associated with the response variable. The criteria used to add a variable is whether it increases the F value for the regression by some specified amount *k*. When a variable reduces F, also by some specified amount *w*, it is removed from the model.

The stepwise regression results show that only the variable TOTKDSI present a beta value significant (beta = 0,671) and F = 14,85257. When we use only the variable TOTOKSI, the *F*-value (used as an overall *F*-test of the relationship between the dependent variable and the set in independent variables) is more strong: F = 50,05215.

Consequently, in this work only TOTKDSI is used to build the ANN and regression model. Future works will involve the COCOMO cost drivers and modes. Likewise, Boehm [7] has shown that the most important predictor for these projects is TOTKDSI – thousands of delivered source instructions.

C. *Training and evaluation*

Estimates of the accuracy of prediction obtained from the training data set are always optimistic. To get a more realistic estimate of the accuracy of prediction we followed the similar procedure as in [11]. Based on this process, we omitted a subset of projects (the test dataset), we next developed a model with the remaining projects (the learning data set), and finally we assessed the predictive accuracy of the model on the test dataset. In this way, we constructed the learning dataset by removing every sixth project starting from the sixth project.

Thus, the learning dataset was constructed by removing the projects 6, 12,18,24,30,36,42,48,54 and 60. Since we used all 63 projects of the COCOMO database in order to build our models, the learning dataset contained 53 projects.

It is to be noticed that each system imposes a set of constraints on data representation. When there are several variables with nominal values in the project database, the data are normalized to fit the interval [0,1]. No normalization was required for the regression analysis.

The neural network was implemented with 1 input, 9 units in the first hidden layer, 4 units in the second layer, and 1 output neuron, using the logistic function. The input variable was TOTKDSI and the neural network was trained to estimate effort IT. The training phase was repeated 15 times, in a search for the best network to solve the problem. Besides, different neural network architectures were tried. But, the results presented in this paper correspond to the neural network with the best generalization performance.

The linear regression model was calibrated using stepwise backward method. After a number of experiments, we achieved a final regression model.

The predictions obtained from the ANN and the regression model (after training on the COCOMO data) using the test dataset are shown in Table 1.

We performed a linear regression/correlation analysis to "calibrate" the predictions, with M_{est} treated as the independent variable and M_{act} treated as the dependent variable. The R^2 value indicates the amount of variation in the actual values accounted for by a linear relationship with the estimated values. R^2 values close to 1.0 suggest a strong linear relationship and those close to 0.0 suggest no such relationship.

TABLE I
REGRESSION AND ANN ESTIMATES

EsforçoIT	Regression	ANN
6,753	26,026	26,5586
59,3	93,678	52,37
3705	673,847	859,7053
120,05	202,332	165,4667
1,12	21,885	25,5485
12,93	48,577	33,0107
10,58	111,104	63,0066
375,24	969,057	638,4523
3,5	26,846	26,7646
9,98	31,561	27,9878

Different error measurements have been used by various researchers, but for this project the main measure for model accuracy is the Mean Magnitude of Relative Error (MMRE) and R^2. MMRE is the mean of absolute percentage errors:

$$MMRE = \frac{\left(\sum_{i=1}^{n} \left| \frac{M_{est} - M_{act}}{M_{act}} \right| * 100 \right)}{n} \quad (2)$$

where there are *n* projects; M_{act} is the actual effort; and M_{est} is the predicted effort.

Others researchers have used the adjusted R squared or the coefficient of determination to indicate the percentage of variation in the dependent variable that is "explained" in terms of the independent variables. In this paper, we have decided to adopt the MMRE and the adjusted squared R as prediction performance indicators since these are widely used.

Table 2 summarizes the MRE and R^2 values resulting from a linear regression of M_{est} and M_{act} values for the stepwise backward regression and the ANN models, and results obtained by Kemerer [24] with COCOMO-Basic, Function Points and SLIM models.

TABLE II
THE PREDICTIVE ACCURACY

	Regress. Eq.	R- square	MMRE
ANN	-1,68+1,676*x	0,85	420
Regression	-1,71+1,623*x	0,83	462
FPA	-37 +0,96x	0,58	103
COCOMO	27,7 + 0,156x	0,70	610
SLIM	49,9 +0,082x	0,89	772

These results indicate that stepwise regression's and ANN's predictions show a strong linear relationship with the actual development effort values for the ten test projects. On this dimension, the performance of the ANN model is less then SLIM's performance in Kemerer's experiments, but better than the stepwise regression models. In terms of MMRE, the ANN performs strikingly well compared to the other approaches, and regression model.

This experiment illustrates two points. In an absolute sense, none of the models perform particularly well at estimating software development effort, particularly along the MMRE dimension, but in a relative sense ANN approach is competitive with traditional models. In general, even though MMRE is high in the case of all models, a high R^2 suggests that by "calibrating" a model's prediction in a new environment, the adjusted model prediction can be reliably used. Along the R^2 dimension, the ANN method provides significant fits to the data.

The authors defend the use of artificial neural networks; witch presents competitive results with other approaches. A primary advantage of ANN is that they are adaptable and nonparametric; predictive models can be tailored to the data at a particular site.

VI. CONCLUSION AND FUTURE WORKS

This paper has compared the neural network method to traditional approaches for software effort estimation. A neural network and a stepwise regression analysis were applied to Boehm's COCOMO dataset in order to predict effort from size. The results of the ANN prediction compare favorably with those obtained from linear regression.

The neural network performed better than linear regression on this data set and we can see why. As illustrated in Figure 2, in this dataset there is one observation with very large efforts that are out of all proportions to their sizes, as well as one with a small effort for its size, and a linear function of size will not be very successful at predicting these. On the other hand, an attempt to solve these outliers could influence negatively in the accuracy regression in accomplishing prediction for other observations. Once the ANN not is limited to a linear function, it can deal more successfully with observations that lie far from the best straight line.

A more homogeneous dataset with no outliers would show the regression method to better advantage. ANN would also perform better on such a dataset.

Although ANN has demonstrated significant advantages in certain circumstances, it does not replace regression and should be regarded as another powerful tool to be used in the calibration of software effort models.

Consequently, new experiments will be conducted to combine the neural network and regression techniques to calibrate and to test prediction models on other datasets, such as, the ISBSG database (*International Software Benchmarking Standard Group*). It contains information on software projects developed with modern software development techniques which permit the impact assessment of current, newer, software development processes (using object-oriented programming languages, C++, Java, legacy code interfaces, validation and verification tools, UML, other requirements documentation, distributed software application, etc.) on the input parameters for software level-of-effort prediction models. The aim is improve the performance of the neural network models obtained in this work and to obtain a model that can safely be used in software development.

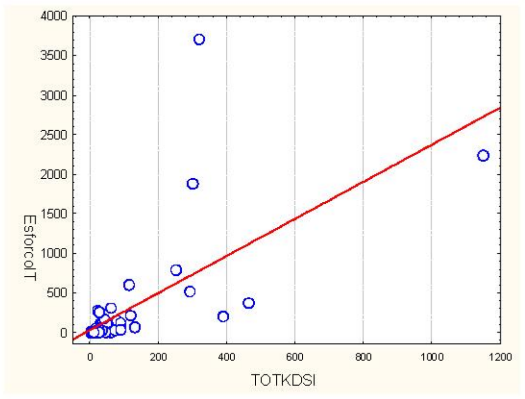

Fig. 2. The COCOMO dataset – effort vs. size

APPENDIX

Following is a brief description of the attributes and their influence on development effort. Refer to Boehm (1981) for a detailed explanation of each attribute.

1. Required Software Reliability (RELY): It measures the reliability of the software.
2. Database Size (DATA): The size of the database to be used by a software system may affect the effort.
3. Product Complexity (CPLX): The application area has a bearing on the software development effort.
4. Execution Time Constraint (TIME): If there are constraints on processing time, the time may be greater.
5. Main Storage Constraint (STOR): If there are memory constraints, then the effort will tend to be high.
6. Virtual Machine Volatility (VIRT). If the underlying hardware and/or software system change frequently, then development effort will be high.
7. Analyst Capability (ACAP): If the analysts working on the software project area highly skilled, the effort of the software will be less than projects with less-skilled analyst.
8 Application Experience (AEXP): The experience of project personnel influences the software effort.
9. Programmer Capability (PCAP): This is similar to ACAP, but it applies to programmers.
10. Virtual Machine Experience (VEXP): Programmer experience with the underlying hardware and the operating system has a bearing on development effort.
11. Language Experience (LEXP): Experience of the programmers with the implementation language affects the software development effort.
12. Personnel Continuity Turnover (CONT): If the same personnel work on the project from beginning to end, then the development effort will tend to be less than similar projects experiencing greater personnel turnover.
13. Modern Programming Practices (MODP): Modern programming practices like structured software design reduces the development effort.
14. Use of Software Tools (TOOL): Extensive use of

software tools like source-line debuggers and syntax-directed editors reduces the software development effort.

15. Required Development Schedule (SCED): If the development schedule of the project is highly constrained, then the development effort will tend to be high.

References

[1] R. Agarval, "Estimating Software projects," Software Engineering Notes, Julho 2001, vol. 26, no 4, pp. 60-57.

[2] C. Jones, *Estimating Software Costs,* McGraw-Hill, 1986.

[3] R. Lai, and S. Huang, "A model for estimate size of a formal communication protocol specification and its implementation," IEEE Transaction on Soft. Engineering, January 2003, vol.29, no 1. pp.. 46-62.

[4] T.E Hasting, and A.S.M. Sajeev, "A vector based approach to software size measurement and effort estimation," IEEE Transactions on Soft. Engineering, April 2001, vol 27., no.4.

[5] L.C. Briand, I. Wieczorek, "Software resource estimation," Encyclopedia of Software engineering, 2002, vol. P-Z, no.2, pp. 1160-1196.

[6] MCT Ministerio da Ciência e Tecnologia, "Qualidade e Produtividade no setor de software," In: http://www.mct.gov.br/Temas/info/Dsi/Quali2001/2001 Tab40.htm, Tabela 40 – Práticas de Engenharia de Software no Desenv. e Manutenção de Software, 2001.

[7] B.W. Boehn, *Software engineering economics*, Prentice-Hall, Englewood Cliffs, NJ, 1981.

[8] A.J. Albrecht, and J.R. Gaffney, "Software function, source lines of code, and development effort prediction: a software science validation," IEEE Transactions on Software Engineering, 1983, vol. 9, no. 6, pp. 639-648.

[9] M. Shepperd, and C. Schofield, "Estimating Software Project Effort Using Analogies," IEEE Transactions on Software Engineering, November 1997, vol.23, no.12, pp.736-743.

[10] Zhong, S.; Khoshgoftaar, T.M.; Seliya, N.: Analysing Software Measurement Data with Clustering Techniques. IEEE Intelligent Systems, 2004, pp.20-27.

[11] P. Sentas, L. Angelis, I. Stamelos, and G. Bleris, "Software productivity and effort prediction with ordinal regression," Journal Information and Software Technology, 2005, no. 47, pp.17-29.

[12] R. Bisio, and F. Malabocchia, "Cost estimation of software projects through case base reasoning," 1st Intl. Conf. on Case-Based Reasoning Research & Development, Springer-Verlag, 1995, pp.11-22.

[13] I. Myrtveit, E. Stensrud, and M. Shepperd, "Reliability and Validity in Comparative Studies of Software Prediction Models," IEEE Transaction on Soft. Engineering, May 2005, vol.31, no. 5, pp. 46-62.

[14] B. Samson, D. Ellison, and P. Dugard, "Software Cost Estimation Using Albus Perceptron (CMAC)," Information and Software Technology, 1997, vol.39, pp. 55-60.

[15] K. Srinivazan, and D. Fisher, "Machine Learning Approaches to Estimating Software Development Effort,". IEEE Transactions on Software Engineering, February 1995, vol.21, no.2, pp.126-137.

[16] A. Albrecht, "Measuring application development productivity," Proc. IBM Application Development Symposium, 1979, pp. 83-92.

[17] W. Boehm, E. Horowitz, R. Madachy, D. Reifer, B.K. Clark, B. Steece, A.D. Brown, C. Abts, *Software Cost Estimation with COCOMOII,* Prentice-Hall, 2000.

[18] S. Vicinanza, M.J. Prietula, and T. Mukhopadhyay, "Case-based reasoning in software effort estimation," In Proc.11th Int. Conf. Info. Syst, 1990, pp.149-158.

[19] N. Karunanitthi, D.Whitley, and Y.K.Malaiya, "Using Neural Networks in Reliability Prediction," IEEE Software, 1992, vol. 9, no.4, pp.53-59.

[20] G. Witting, and G. Finnie, "Using Artificial Neural Networks and Function Points to Estimate 4GL Software Development Effort", J. Information Systems, 1994, vol. 1, no. 2, pp. 87-94.

[21] T. M. Khoshgoftaar, E.B. Allen, and Z. Xu, "Predicting testability of program modules using a neural network," Proc. 3rd IEEE Symposium on Application-Specific Systems and Sof. Eng. Technology, 2000, pp. 57-62.

[22] G. Witting, and G. Finnie, "Estimating software development effort with connectionist models," Inf. Software Technology, 1997, vol. 39, pp. 369-476.

[23] P. Kok, B.A. Kitchenham, J. Kirakowski, "The MERMAID approach to software cost estimation," Espirit Technical Week, 1990.

[24] C.F. Kemerer, "An empirical validation of software cost estimation models," Comunication of ACM, May 1987, vol.30, pp.416-429.