SOFTWARE EFFORT ESTIMATION: THE ELUSIVE GOAL IN PROJECT MANAGEMENT

J. Javier DOLADO, Luis FERNÁNDEZ*, M. Carmen OTERO and Leire URKOLA
University of the Basque Country, 20.009 - San Sebastián, Spain

Work supported by UPV-EHU 141.226-EA083/98

**European University of Madrid, Villaviciosa de Odón, Madrid, Spain*

Key words: Effort Estimation, Project Management, Genetic Programming, Neural Networks, Linear Regression

Abstract: The estimation of the effort to be spent in a software project is a problem still open. Having a good estimation of the variables just at the beginning of a project makes the project manager confident about the future course of the actions, since many of the decisions taken during the development depend on, or are influenced by, the initial estimations. The root of the problems can be attributed to the different methods of analysis used, and to the way with which they are applied. On one hand we may not use the adequate independent variables for prediction and/or we may not build the correct predictive equations. On the other hand we could think that the method of prediction has some effect on the predictions, meaning that it is not the same to use classical regression or other methods of analysis. We have applied linear regression, neural networks and genetic programming to several datasets. We infer that the problem of accurate software estimation by means of mathematical analysis of simple relationships solely isn't going to be inmediately solved.

1. THE ESTIMATION OF SOFTWARE COST

After several decades of research into software management, the question of how to predict effort with sufficient reliability is still unsolved. The prediction of the effort to be spent in a project is the most sought variable by the project manager. However, finding those good initial estimations has been shown to be elusive. In fact, and after three decades of research, the precise function that relates effort and size is still unknown. Different equations represent different hypotheses about the economies of scale in software development. The type of economy of scale of a production process indicates how the output of a process responds to the increments of the inputs to the own process. Therefore, the manager can take appropriate actions if he/she knows how much software has to develop and he/she also knows how is the equation of the effort with respect to the size of the project. To identify the cost function is still an unsolved question.

The root of the problems can be attributed to the different methods of analysis used and to the way which they are applied. On one hand we may not use the adequate independent variables for prediction and/or we may not build the correct predictive equations. On the other hand we could think that the method of prediction has some effect on the predictions, meaning that it is not the same to use classical regression or other methods of analysis. In this work we discuss the predictive capabilities of some of the estimation models and methods expounded in the literature: a) with respect to the type of size used as independent variable, b) with respect to the sample used and c) with respect to the type of method of analysis.

The best known models, such as COCOMO, have been developed using classical regression as the main method for building the equations. Newer methods such as neural networks have gained use among estimators. Other methods include the use of case-base reasoning and "expert judgement". Recently, and for the purposes of system identification in chemical process engineering, the technique of “genetic programming” has been used with promising results. The underlying idea is to mimic the biological process of “natural selection”, in such way that the best solution is achieved by probabilistic evolution. It is not guaranteed that the best solution is to be found, but only that one of the best solutions will appear. This is a type of the current “evolutionary computation” techniques which are able to cope with “hard problems”. The use of the new technique of genetic programming is appealing since it has the flexibility of adjusting a variety of functions to the data points.

Therefore, the main theme of research of this work is to examine the methods with which we can predict effort. Also we will compare the predictive capabilities of each approach. When applying a method other assumptions are tested, such as the use of different samples on the same datasets and the effect of some outliers. In this article the data of the works of (Abran and Robillard 1996), (Miyazaki et al. 1994), and (Matson et al. 1994) are analysed. We especially analyse the method of GP compared to the other methods, since GP is a type of evolutionary computation technique that has some advantageous properties over classical regression. GP can be an alternative to classical symbolic regression since it can include many constraints in the process of deriving the equation.

2. DIFFERENT METHODS OF ANALYSIS

The main method used in the literature for deriving equations A comparison of alternative methods to the classical linear regression can be found in (A.R. Gray and S.G. MacDonell 1997). Next we state the methods used in the present work.

2.1 Statistical Analysis by Linear Regression

Linear regression (LR) is one of the classical technique for analysing and building estimation models. It is assumed that the dependent variable is linearly related with the independent(s) variable(s). In order to have other functions or equations, several transformations can be made in the variables. Usual transformations take the logarithm of the variables. Goodness of fit of the model is usually measured by the coefficient of multiple determination R^2, and by the mean squared error. Outlier analysis is also a requirement for asserting the validity of a model.

Questions have been raised in some works about the use of samples for building a model. It is usually assumed in the statistical field, and also in the software engineering field, that we obtain different results if we choose a subset or the whole dataset. Therefore, two possibilities will be taken into account in the analysis that is carried out next: to build a model with all points in the dataset or to use a subset for model building and the rest of the points for model evaluation. And, since the sample can also affect the results, we have used three different samples, randomly selected, for building models. The subsets are composed of 66% of the cases for model building and 34% for model evaluation. The same applies to the other methods, neural networks and genetic programming.

2.2 Neural Networks

We have also used neural networks for approximating the relationship effort-size. Neural networks (NN) are nets of processing elements that are able to learn the mapping existent between input and output data It has been proved that the NN are universal approximators and that a two-level NN is able to approximate any function. They have already been used previously in the software engineering field. The appeal of using NN is that if there is an underlying relationship in the data under study, a neural net should be able to find it. The drawback of this method lies in the fact that the analyst can't manipulate the net once the learning phase has finished. NN in this work function as nonlinear regression models (Finnie et al. 1997).

2.3 Genetic Programming

The genetic programming technique is a type of evolutionary computation. It was proposed by Koza (Koza 1992), after the author tried to automatically evolve lisp-programs. Evolutionary computation techniques are characterised by the fact that the solution is achieved by means of a cycle of generations of candidate solutions that are pruned by the criteria 'survival of the fittest'. Genetic programming (GP) has been used in a variety of fields, including the automated synthesis of circuits, nonlinear system identification in chemical process engineering (identifying relevant variables), symbolic regression, etc. (Bäck et al. 1997). Here, genetic programming is used as an automated symbolic regression to derive the equations.

The name *genetic programming* comes from the similarities with the paradigm of 'natural selection' in the biological area. There, schematically, populations of specific species evolve according to random mutations of the genes and to the 'appropriateness' to the environment of the species. In the case of system identification (symbolic regression) the idea is also simple: to generate randomly a set of initial equations that relate the input and the output variables and select the equations according to the principle 'survival of the fittest'. The process is repeated until an acceptable solution is obtained.

GP is a nonparametric method since it does not make any assumption about the distribution of the data, and derives the equations according only to fitted values. The use that is made here can be considered as an alternative to Curve Estimation (and Linear Regression), and will also allow us to compare the linear equations with those derived automatically. Since a set of different equations are derived in different runs, only those that give the best results in the evaluation subset are reported.

The equation compared in the present work is the one that gives the best fit in the last generation. This equation is applied to the evaluation data. Each run had an initial population of 25 to 50 randomly generated equations. The number of generations in each run varied, but the best results were obtained with 3 to 5 generations. The number of equations that remained from one generation to the next represented 10% of the previous one. The new equations of the new generation were formed by:

a) *crossover*: two equations exchange parts, preserving the syntax of the mathematical expression;

b) *mutation*: a term of the equation (function, variable or constant) is changed randomly.

In general, GP has worked as expected in providing equations "fitted to the data". The algorithm that evolves a solution for symbolic regression is described below. The software for GP has been provided by the Newcastle Symbolic Optimisation Group (runs in Matlab 4.2.c) (Willis et al. 1997) (McKay et al. 1997).

Genetic Programming Algorithm

```
1. Generate initial population of
equations
2. While there are generations to run
do
     2.1. Evaluate fitness of each
     equation
     2.2. For each equation in the
     population select randomly one of
        a) mutation with probability Pm
        b) crossover with probability Pc
        c) direct reproduction with
        probability (1-Pm-Pc)
     2.3. Add the new equation to the
     new population
     Endfor
Endwhile
```

Each equation in the population is represented as a tree. The operation of *crossover* takes a subtree of two equations and exchanges them to form new members of the populations. *Mutation* randomly modifies a subtree of an equation.

The functional set used is: +, -, *, /, ^, sqrt, square, log, exp. Probabilities of mutation and crossover are in the range of Pm=0.8 and Pc=0.2. The fitness measure in the present work is the *mean squared error* of the predicted values of the equation.

2.4 Evaluation of the techniques for software cost estimation

The criteria for comparing the three methods on the datasets is to compute the values for PRED(0.25) and MMRE ≤ 0.25 (Conte et al. 1986). In the case of multiple linear regression an assessment is made by the R^2 and other statistical criteria, in order to select the best equation. However, we consider that the most reasonable variables are PRED(0.25) and MMRE, since they measure the capability of prediction, not the statistical explanation.

Mean magnitude of relative error, (MMRE), is defined as MMRE = $1/n \cdot \sum|(e_i-\hat{e}_i)/e_i|$, where *e* is a real value of a variable in a project, *ê* is its estimate and *n* is the number of projects. Thus if the MMRE is small, then we have a good set of predictions. A usual criteria for accepting a model as good is that the model has a MMRE ≤ 0.25.

Prediction at Level l (PRED(l)), where *l* is a percentage, is defined as the quotient of number of cases in which the estimates are within the *l* absolute limit of the actual values divided by the total number of cases. For example PRED(0.1) = 0.9 means that 90% of the cases have estimates within the 10% range of their actual values. A standard criteria for considering a model as acceptable is PRED(0.25) ≥ 0.75. This means that at least 75% of the estimates are within the range of the 25% of the actual values.

3. DATASETS FOR ANALYSIS

In the three datasets that follow the dependent variable is effort, measured in person-months or person-days. We will consider only one independent variable, although in some analysis several variables have been used.

3.1 DATASET 1

This set of 21 data points was collected by (Abran and Robillard 1996). The independent variable is “unadjusted function points”.

3.2 DATASET 2

This dataset is composed of 47 projects and was collected by (Miyazaki et al. 1994). The independent variable is “thousands of LOC”. This dataset was used to compare the method of “robust regression” to the least squares criteria.

3.3 DATASET 3

This dataset is an approximation to those used by (Matson et al. 1994). Since the original data was not completely available, an approximation has been achieved by using a recovery procedure from the original figures. Therefore results must be taken with caution, although the high number of points (104) can moderate the variations.

4. APPLICATION OF THE METHODS TO THE DATASETS

Below are the results of applying the three methods (genetic programming, neural networks and linear regression) to the datasets. Two types of analysis have been applied: a) to use the same data points for model building and model evaluation, and b) to use a sample for model building and the rest of the points for model evaluation; in this case three different samples have been used, in order to examine the effect of the sample.

4.1 Analysis with all data points in each dataset

The results when using all data points are as follows:

Dataset 1
Genetic Programming
PRED(0.25) = 81; MMRE = 0.221
Neural Networks:
PRED(0.25) = 80.95; MMRE = 0.279
Curve Estimation:
PRED(0.25) = 57.14; MMRE = 0.2339

Dataset 2
Genetic Programming
PRED(0.25) = 40.4; MMRE = 0.643
Neural Networks
PRED(0.25) = 40.43; MMRE = 0.5886
Curve Estimation
PRED(0.25) = 42.55; MMRE = 0.3999

Dataset 3
Genetic Programming
PRED(0.25) = 27.9; MMRE = 1.031
Neural Networks
PRED(0.25) = 30.77; MMRE = 1.1448
Curve Estimation
PRED(0.25) = 27.88; MMRE = 0.8485

We observe that GP obtains slightly better values in the dataset 1. The NN obtains the best result in dataset 3. Linear regression works better than the two other methods in dataset 2. But, except for dataset 1, the predictive values are very far from the ideals. The predictions in dataset 1 could be accepted by genetic programming. In dataset 2 the different methods do not seem to have much impact on the results. Dataset 3 obtains bad results, regardless of the method.

4.2 Results using a subset of the data for model building and evaluation

4.2.1 DATASET 1

SAMPLE 1
Genetic Programming
PRED(0.25) = 100; MMRE = 0.141
Neural Networks:
PRED(0.25) = 100; MMRE = 0.1364
Curve Estimation:
PRED(0.25) = 50; MMRE = 0.2727

SAMPLE 2
Genetic Programming
PRED(0.25) = 87.5; MMRE = 0.158
Neural Networks
PRED(0.25) = 87.5; MMRE = 0.1422
Curve Estimation
PRED(0.25) = 62.5; MMRE = 0.2208

SAMPLE 3
Genetic Programming
PRED(0.25) = 62.5; MMRE = 1.086
Neural Networks
PRED(0.25) = 62.5; MMRE = 0.6612
Curve Estimation
PRED(0.25) = 25; MMRE = 0.6776

In this analysis we observe how the sample has a great effect on the results, and that GP and NN seem to work equally well. Results in the samples are better than the complete dataset.

4.2.2 DATASET 2

SAMPLE 1
Genetic Programming
PRED(0.25) = 50; MMRE = 0.494
Neural Networks:
PRED(0.25) = 56.25; MMRE = 0.3403
Curve Estimation:
PRED(0.25) = 37.5; MMRE = 0.3452

SAMPLE 2
Genetic Programming
PRED(0.25) = 50; MMRE = 0.69
Neural Networks

```
PRED(0.25) = 43.75; MMRE = 0.713
Curve Estimation
PRED(0.25) = 31.25; MMRE = 0.8411
```

SAMPLE 3

```
Genetic Programming
PRED(0.25) = 50; MMRE = 0.457
Neural Networks
PRED(0.25) = 56.25; MMRE = 0.3496
Curve Estimation
PRED(0.25) = 37.5; MMRE = 0.2911
```

Sampling with different data points does not have much effect on the results. GP and NN obtain the best results. Classical regression produces bad results in all cases. Similarly to paragraph 4.1 the results are not very impressive, regardless of the sample. However, the samples obtain better results than using all data points.

4.2.3 DATASET 3

SAMPLE 1

```
Genetic Programming
PRED(0.25) = 31.4; MMRE = 1.199
Neural Networks:
PRED(0.25) = 31.43; MMRE = 1.1381
Curve Estimation:
PRED(0.25) = 28.57; MMRE = 0.8363
```

SAMPLE 2

```
Genetic Programming
PRED(0.25) = 31.4; MMRE = 0.485
Neural Networks
PRED(0.25) = 25.71; MMRE = 0.6045
Curve Estimation
PRED(0.25) = 25.71; MMRE = 0.494
```

SAMPLE 3

```
Genetic Programming
PRED(0.25) = 28.6; MMRE = 1.029
Neural Networks
PRED(0.25) = 31.43; MMRE = 1.0621
Curve Estimation
PRED(0.25) = 20; MMRE = 0.804
```

Although NN and GP work better than classical regression, the results are bad in all cases. Therefore, sampling has not improved over the whole dataset.

4.3 The methods

We have seen that both genetic programming and neural networks have the capability of predicting the effort-size relationship similarly to curve estimation, at the least. In other cases GP behaves better, since it also provides an equation. The drawback of the NN lies in the fact that although it predicts well, no other hint is provided to the human estimator about possible improvements in the function.

Ideally, several methods should be used to make predictions, because each one has its benefits.

4.4 Sampling versus whole datasets

We observe from the data provided above that the usual assumption about the use of all data points is not true. We have not obtained better values in section 4.1 than in section 4.2, at least in the datasets analysed here. However, generalizations should be made with caution.

5. CONCLUSIONS

5.1 Effort Estimation

The first conclusion that we can draw from the values obtained is that it is difficult to obtain good results in the process for effort estimation, as the PRED(0.25) and MMRE values evidence. The quality of the data seems more important than the method used. This may also indicate that more variables (or other factors) are needed in order to quantify effort. A simple measure of size may not be enough for making predictions.

Although we do not provide the equations here, there has not been any pattern in the form of the equations as to generalize the effort-size relationship. The results show that algorithmic estimation is only acceptable in some cases.

Therefore, there remains the problem of how to build the model starting from the data points. It is not clear that the use of the sample should be advisable.

But, the underlying question is described as to what the estimators want. If we want making predictions without precisely knowing the equation then the paradigm of NN may be appropriate. On the contrary, if we need a symbolic equation then we have to resort to the linear regression or to the genetic programming paradigms. GP has great flexibility and power, therefore providing many benefits to the process of estimation.

5.2 Size Estimation

It can be noticed that one of the first estimations that are made at the beginning of a project is that of size estimation. We observe that the most used measures are LOC and function points. Therefore, we could have the temptation to ask ourselves: does the problem lie in the independent variable "size"?

In the results of the above sections we have obtained good results for function points in one case and very bad in the other, and moderate results for LOC. However, there are cases that have found no relationship at all between effort and function points (Dolado 1997). And further results have proved that function points are only the *de facto* measure for functional software size, but that there are other possibilities such as the *component-based method* for software size estimation (Dolado 1999).

We see that the process of making estimations may be constrained by the ways the data is collected, and that new ways of estimating should be researched in order to overcome the limits of purely algorithmic methods.

ACKNOWLEDGMENTS

This work has been partially supported by project UPV-EHU 141.226-EA083/98. The authors are also grateful to the Newcastle Symbolic Optimization Research Group, who provided the software for genetic programming.

REFERENCES

A. Abran and P.N. Robillard, "Function Point Analysis: An Empirical Study of Its Measurement Processes", IEEE Trans on Software Engineering, Vol. 12, No. 12, December 1996, pp. 895-910

T. Bäck, U. Hammel, and H.-P. Schwefel, "Evolutionary Computation: Comments on the History and Current State", IEEE Trans. on Evolutionary Computation, 1 No.1, April 1997, pp. 3-17

S.D. Conte, H.E. Dunsmore and V.Y. Shen, Software Engineering Metrics and Models, Benjamin/Cummings, 1986

J.J. Dolado, "A Validation of the Component-Based Method for Software Size Estimation", IEEE Trans. on Software Engineering, to appear, 1999

J.J. Dolado, " A Study of the Relationships among Albrecht and Mark II Function Points, Lines of Code 4GL and Effort", Journal of Systems and Software, Vol. 37, pp 161-173, May, 1997

G.R. Finnie, G.E. Wittig and J-M. Desharnais, "A Comparison of Software Effort Estimation Techniques: Using Function Points with Neural Networks, Case-Based Reasoning and Regression Models", Journal of Systems and Software, Vol. 39, pp. 281-289, 1997

A. R. Gray and S. G. MacDonell, "A comparison of techniques for developing predictive models of software metrics", Information and Software Technology, 39, pp. 425-437, 1997

J.R. Koza, Genetic Programming: On the Programming of Computers by Natural Selection, MIT Press, 1992

J.E. Matson, B.E. Barret and J.M. Mellichamp, "Software Development Cost Estimation using Function Points", IEEE Trans. on Software Engineering 20, No. 4 April, pp. 275-287, 1994

B. McKay, M.J. Willis and G.W. Barton, "Steady-State Modelling of Chemical Process Systems using Genetic Programming", Computers and Chemical Engineering, 1997, 21 (9) pp. 981-996

Y. Miyazaki, M. Terakado, K. Ozaki and H. Nozaki, Robust Regression for Developing Software Estimation Models, Journal of Systems and Software, Vol. 27, pp. 3-16, 1994

M.Willis, H. Hiden, M. Hinchliffe, B. McKay and G. Barton, "Systems Modelling Using Genetic Programming", Computers and Chemical Engineering, 1997, Vol. 21 Supplement, pp. 1161-1166.