Full citation: MacDonell, S.G., & Gray, A.R. (1998) A comparison of modeling techniques for software development effort prediction, in Proceedings of the Fourth International Conference on Neural Information Processing (ICONIP'97/ANZIIS'97/ANNES'97). Dunedin, New Zealand, Springer-Verlag, pp.869-872.

A Comparison of Modeling Techniques for Software Development Effort Prediction

S.G. MacDonell and A.R. Gray

Department of Information Science, University of Otago
PO Box 56, Dunedin, New Zealand
Email: stevemac@commerce.otago.ac.nz

Abstract

Software metrics are playing an increasingly important role in software development project management, with the need to effectively control the expensive investment of software development of paramount concern. Research examining the estimation of software development effort has been particularly extensive. In this work, regression analysis has been used almost exclusively to derive equations for predicting software process effort. This approach, whilst useful in some cases, also suffers from a number of limitations in relation to data set characteristics. In an attempt to overcome some of these problems, some recent studies have adopted less common modeling methods, such as neural networks, fuzzy logic models and case-based reasoning. In this paper some consideration is given to the use of neural networks and fuzzy models in terms of their appropriateness for the task of effort estimation. A comparison of techniques is also made with specific reference to statistical modeling and to function point analysis, a popular formal method for estimating development size and effort.

1. INTRODUCTION

As software development has become a more and more crucial investment for many organizations there has been a greater awareness of the need to better model the development process. The financial gains available as a result of better project management are considerable for many organizations, with even slight improvements in predicting the almost-chaotic dynamics of software development appreciated. Models are therefore needed for the purposes of predicting, monitoring, controlling, and assessing software development. In this paper the emphasis is on the goal of predicting development effort, one of the most widely researched areas.

Predictive models of the software development process are of considerable interest to a wide range of stakeholders, including the software users, customers contracting for development, contractors bidding for development, developers, and managers in general. Such models are generally constructed using software metrics as the dependent and independent variables.

Software metrics are measurements concerning either the software being developed (the product) or the manner in which it is being developed (the process) [Fenton, 1991]. Examples of product metrics would be the size of a system (perhaps in terms of the number of lines of code or a functionality based measure, such as the number of screens and reports), the number of defects in a system remaining after testing, or the complexity of a module (defined in some manner). Examples of process metrics would include the number of developers working on the project, the effort required for various stages of development, and the experience of the developers.

Traditionally, such metrics were used as part of either a formally specified model (such as Function Point Analysis [Garmus and Herron, 1995]) which could be calibrated to a specific organization and/or environment, or they were used as variables in a regression equation. For example, the effort (number of programmer hours) required for testing a particular series of modules might be the dependent variable in a model with a functionality measure of size, complexity, and developer experience as independent variables.

The field of software metrics has become a well-researched area and while such models have provided moderate success in the past, a number of concerns have arisen. These include the following:

1. The accuracy of the models themselves. Linear models without interactions have been traditionally used in the interests of parsimony. While transformations, usually exponential, have also been used to compensate for non-linearities, the effect on the objective function (such as Least Squares) can make predictions less than optimal. Nonlinear regression has rarely been used, largely due to a desire for simplicity and also a lack of sufficient quantities of data.
2. The ability to incorporate expert knowledge into a model with the intention of reducing the free parameters to compensate for the small quantities of

data that are usually available. The use of regression has allowed only some very limited expert knowledge to be used in the process of model development.

3. The requirement for exact values of independent variables to be provided to the currently used models. Almost all software metric models are based on an *exact* values in and *exact* values out approach.

4. Related to Point 3 above, the fact that model outputs are exact values often leads to an overcommittment to these values, with the associated risks for poor accuracy magnified. It would therefore seem desirable if the outputs of such models could be expressed as *fuzzy* variables, with more precise predictions made as the development process advances and additional information becomes available.

5. The small quantities of data that are generally available for software metrics is limiting in terms of the techniques that can be used. While on some occasions, larger data sets are available, that would permit the use of non-linear statistical and neural network models, in the majority of cases more parsimonious models must be developed. Reasons for the small size of data sets include the fact that the number of variables influencing the process is so large that a considerable number of observations would be required before that data set could be considered more than small. Large data sets are generally collections of data from a number of organizations, which makes generalization to a particular organization even more difficult. The constantly changing dynamics of the development process, in terms of new tools and development methodologies, also makes the gathering of relevant past data difficult.

Once the best-fit model has been determined (by whatever method), its consistency and accuracy can be assessed by using a validation data set. The use of some data set that has not exerted any influence whatsoever on the model development and selection is essential for an unbiased estimate of the model's generalization capabilities. In the case of software metrics, the holdout error is crucial for assessing the risk associated with the model's predictions.

Many different methods for estimating a model's fit are available. These include the many forms of correlation (R^2, adjusted R^2, R^2 adequate), Akaike Information Criterion, and many others. A set of indicators is commonly used in metrics analysis to indicate the adequacy of a predictive model; namely the mean magnitude of relative error (MMRE) and the threshold-oriented pred measure.

The magnitude of relative error (MRE) is a normalized measure of the discrepancy between the actual data values (V_A) and the fitted values (V_F):

$$MRE = \frac{|V_A - V_F|}{V_A} \quad (1)$$

The mean MRE (MMRE) is therefore the mean value for this indicator over all observations in the sample. A lower value for MMRE generally indicates a more accurate model from the perspective of a project manager.

The pred measure provides an indication of overall fit for a set of data points, based on the MRE values for each data point:

$$pred(l) = \frac{i}{n} \quad (2)$$

In equation (2) l is the selected threshold value for MRE (from equation (1)), i is the number of data points with MRE less than or equal to l, and n is the total number of data points. As an illustration, if pred(0.20) = 30%, then we can say that 30% of the fitted values fall within 20% of their corresponding actual values. In terms of assessing the performance of a given model, contemporary expectation of a *good* model is the achievement of pred(30) = 60%.

The comparison of modeling techniques that follows is based on the analysis of a set of more than eighty systems development observations collected over a period of four years [Desharnais, 1989]. This data is of the form often encountered in software metrics, a number of project characteristics to be used to predict development effort. The specific details of the systems are beyond the scope of this paper, where the emphasis is on evaluating modeling techniques rather than developing actual models to be used for similar projects. The interested reader is therefore referred to [Desharnais, 1989] for further details.

The data set includes measurements of project effort, project duration, levels of experience with development equipment and in project management, numbers of basic transactions and data entities, and the raw and adjusted function point counts. For a more extensive and theoretical comparative review of such techniques applied to software metric models see [Gray and MacDonell, 1997].

The first analysis is linear regression analysis under the Least-Squares and Least-Median-Squares approaches. Next, results are shown for analysis based on a feedforward neural network model. This is followed by a discussion of how a fuzzy logic model could have been used for even earlier estimation. Numerical results are not provided for the fuzzy logic technique, since its application is most suited to early project estimation and the required data is not available.

A total of 81 observations are available from the data set. Each modeling scenario has used a randomly selected set of fifty-four observations for model construction (training, and testing where appropriate), leaving a validation set of twenty-seven observations.

2. REGRESSION MODELS

Linear regression under the Least-Squares model attempts to find the line that minimizes the sum of the squared errors. Regression analysis is commonly preceded by the creation of two-dimensional scatter plots and exploratory correlation analysis in order to first intuitively, as well as quantitatively, determine the potential relationships that may exist in the data. It is important to keep in mind that the linear nature of such regression only refers to the linear form of the coefficients. Transformations can be used in advance on variables to permit non-linear modeling providing the appropriate transformation is known. This does however alter the meaning of the objective function if the dependent variable is subject to transformation, which may lead to non-optimal prediction. In such cases, nonlinear regression is preferred. Similarly interaction effects can be simulated by the creation of one or more new variables appropriately defined. The primary advantage of this technique is that it is well known to, and understood by, both software metricians and project managers.

Problems arise in relation to the use of least-squares linear regression due in part to the fact that the method assumes a reasonably normal underlying data distribution. All too frequently, however, data sets derived from software engineering do not adhere to this assumption – data is often highly skewed, containing a number of outlier values relative to the number of observations [Kitchenham and Pickard, 1987]. For instance, module size data tends be significantly skewed to the right due to the influence of a few very large modules, whilst the majority of values cluster around a 'standard' size (due to organizational standards and overhead code). In such 'non-normal' cases, the least-squares regression model loses much of its efficiency [Hampel *et al*., 1986; Myrvold, 1990].

This problem of analysis can be at least partially overcome through the application of the less common Least-Median-Squares regression technique. This *robust* approach determines outlier values prior to final regression, and enables the analyst to discard or weight appropriately the outlier observations. By minimizing the median squared error, the method is robust to data contamination of up to fifty-percent. Thus the main body of observations remains integral to the development of the relationship whilst outlier observations, which may be questionable in terms of reliability or accuracy, can be treated more appropriately. The result is generally a more robust predictive model, particularly in the case where the data set concerned is small, as is often the case in metrics analysis.

3. NEURAL NETWORK MODELS

Neural networks have been applied to software metric modeling in a large number of studies including those described in [Karunanithi *et al*., 1992; Kumar *et al*., 1994, Srinivasan and Fisher, 1995; Wittig and Finnie, 1994]. The results have, in general, been favorable to this particular technique where sufficiently large data sets have been available.

Multi-Layer Perceptron (MLP) networks were developed using two-thirds of the 54 observations for training, and one-third for a testing set. Training was stopped when the testing error was minimized, and the lowest testing error was used to select the particular network architecture with performance as shown in Table 1. This table provides the performance of the best network (on the testing data) for all three data sets, in terms of the three most common measures of accuracy in software metric modeling.

Table 1. Results for the Best MLP Model

	Training Data	Testing Data	Validation Data
Pearsons Correlation	0.8896	0.7745	0.7379
MMRE	0.2968	0.4586	0.43508
pred(10)	6/35	1/19	7/27
pred(25)	18/35	7/19	17/27
pred(50)	31/35	15/19	20/27

It is noted here that the performance of the network is not overly impressive, with 7 out of the 27 validation cases not being predicted even within 50%. The validation errors shown here provide realistic estimates of how the model would perform if used in real-world project management, rather than as an academic after-the-fact analysis.

4. FUZZY LOGIC MODELS

While fuzzy logic models were not developed for this data set, a number of points regarding the use of such models for software metrics are made here.

The use of fuzzy logic models for software metric modeling seems to be appropriate for using the existing expert knowledge available from developers and managers. The vast majority of organizations use some form of expert-judgement as part of their project planning process. This knowledge tends to generalize as techniques and tools change, while numerical data is difficult to recalibrate.

Fuzzy logic also provides a less harsh form of commitment. A project manager may specify that a project will have a *large number of entities*, a *small number of files*, and similarly the other variables. These can be represented as fuzzy variables and a series of rules can then be used to derive some prediction for the output, in this case the project effort. This effort measure could be defuzzified into a number, or left as a slightly vague linguistic label in order to encourage the idea that this is only an estimate.

5. COMPARISON OF TECHNIQUES

It is not sufficient to merely select between techniques based on model accuracy. Other issues such as usability, representation of uncertainty, data requirements, and

meaningfulness of the model itself are also of considerable importance. However, numerical accuracy is here used as an initial assessment criterion.

In order to compare the techniques in terms of their predictive accuracy, a standard metric model for development effort prediction was added to the analysis. One such standard is that of function point analysis (FPA) which is the method of choice in system sizing and effort estimation activities for many large organizations. FPA provides a well-established method for the relatively early assessment of system scope, based on various transaction-oriented system requirements characteristics.

As can be seen in the results presented in Table 2, the most accurate model in terms of MMRE is by far the neural network model. This is to a large extent due to the non-linearities and interactions present in the data, which is barely large enough for such features to be taken into account with regression analysis. However, in terms of classification accuracy, the neural network model is fairly comparable to the Least-Squares regression model after outlier removal from the training and testing data based on residual analysis.

Table 2. Overall Results for the Validation Data

Method	MMRE	pred(10)	pred(25)
FP estimation (mean-based)	0.70	4%	22%
FP estimation (median-based)	0.89	19%	41%
LS regression	0.86	15%	41%
LS regression (no outliers)	0.88	30%	56%
LMS regression	0.85	7%	41%
Neural network	0.44	26%	63%

Clearly these performance indicators are not in themselves very encouraging - one would hope for much more accurate predictions in order to effectively manage the development process. The objective of this study, however, was to compare a selection of analysis methods using the same data set, so as to emphasize the potential of the various analysis options and their capacity to provide effective general models for estimation.

6. CONCLUSIONS AND FURTHER RESEARCH

This paper has illustrated some of the advantages that may be gained when a variety of data analysis methods are considered and the most appropriate method chosen for the development of predictive models. Traditional approaches to development effort estimation may be augmented by such methods as neural networks and fuzzy logic in order that the greatest possible use can be made of whatever data and knowledge is available. When combined with site-based model calibration, there is significant potential for more effective estimation.

In terms of further investigation, our work is continuing in the use of fuzzy logic models, neuro-fuzzy hybrids, and case-based reasoning as other data analysis approaches. Other areas of interest include the consistency of experts' classification of projects in terms of fuzzy logic, and the psychological effect of using various modeling techniques on the users of the models.

The other major focus continuing from this work is the development of a paradigm for selecting the most appropriate technique for modeling software metric models. This is not merely a matter of selecting the technique with the greatest mapping, or generalization, capability as was discussed earlier in the paper.

Preliminary results suggest that neuro-fuzzy hybrids [Kasabov et al., 1997] may be used in many cases to real effect in producing robust, generalisable and intuitively appealing estimation models.

REFERENCES

[Desharnais, 1989] J-M Deharnais, *Analyse statistique de la productivitie des projects de development en informatique apartir de la technique des points des fontion*, Master's Thesis, Universite du Montreal, 1989

[Fenton, 1991] N.E. Fenton, *Software metrics - a rigorous approach*, London UK, Chapman & Hall, 1991

[Garmus and Herron, 1995] D. Garmus, and D. Herron, *Measuring the software process: a practical guide to functional measurement*, Englewood Cliffs NJ, USA, Prentice Hall, 1995

[Gray and MacDonell, 1997] A.R. Gray, and S.G. MacDonell, A comparison of model building techniques to develop predictive equations for software metrics, *Information and Software Technology*, to appear, 1997

[Hampel *et al*., 1986] F.R. Hampel, E.M. Ronchetti, P.J. Rousseeuw, and W.A. Stahel, *Robust statistics*, New York NY, USA, John Wiley & Sons, 1986

[Karunanithi *et al*., 1992] N. Karunanithi, D. Whitley, and Y.K. Malaiya, Prediction of software reliability using connectionist models, *IEEE Transactions on Software Engineering*. 18, 563-574, 1992

[Kasabov et al., 1997] N. Kasabov, J.S. Kim, M. Watts, and A. Gray, FuNN/2 - A fuzzy neural network architecture for adaptive learning and knowledge acquisition, *Information Sciences: Applications*, to appear, 1997

[Kitchenham and Pickard, 1987] B. Kitchenham and L. Pickard, Towards a constructive quality model part ii: statistical techniques for modeling software quality in the esprit request project, *Software Engineering Journal* 2(4): 114-126, 1987

[Kumar *et al*., 1994] S. Kumar, B.A. Krishna, and P.S. Satsangi, Fuzzy systems and neural networks in software engineering project management, *Journal of Applied Intelligence* 4, 31-52, 1994

[Myrvold, 1990] A. Myrvold, Data analysis for software metrics, *Journal of Systems and Software* 12: 271-275, 1990

[Srinivasan and Fisher, 1995] K. Srinivasan and D. Fisher, Machine learning approaches to estimating software development effort, *IEEE Transactions on Software Engineering* 21, 126-137, 1995

[Wittig and Finnie, 1994], G.E. Wittig and G.R. Finnie, Using artificial neural networks and function points to estimate 4GL software development effort, *Australian Journal of Information Systems* 1(2), 87-94, 1994