On configuring a case-based reasoning software project prediction system

Gada Kadoda, Michelle Cartwright, and Martin Shepperd
Empirical Software Engineering Research Group
Department of Computing
Bournemouth University
Talbot Campus
Poole, BH12 5BB, UK
Email: {gkadoda, mcartwri, mshepper}@bournemouth.ac.uk

September, 2000

Abstract
This paper explores some of the practical issues associated with the use of case-based reasoning (CBR) or estimation by analogy for software project effort prediction. We note that different research teams have reported widely differing results with this technology. Whilst we accept that underlying characteristics of the datasets being used play a major role we also argue that configuring a CBR system can also have an impact. We examine the impact of the choice of number of analogies when making predictions. Our analysis is based on project effort data derived from two sources. The first data has been collected by a Canadian software house. The other data was obtained by means of simulation. The results show that using a larger number of analogies generated better predictions. On the other hand, the nearest neighbour is observed to be the better predictor for smaller training sets. We also found that the underlying distribution of the dataset does have a significant impact on the choice of the number of analogies. It was difficult to identify patterns for datasets with complex distribution properties. We believe that a considerable amount of work is required to understand the relationship between the properties of the dataset and configuring a CBR system.

Keywords: Software project, effort prediction, cost estimation, case based reasoning, analogy

1. Background

The software development community acknowledges that timely, accurate estimates of development effort are important to the success of major software projects. Estimates are used when tendering bids, for evaluating risk, resource scheduling and progress monitoring. Inaccurate estimates have implications for all these activities, and so, ultimately, the success or failure of a project. Many projects fail or run over budget. Examples include the MoD's Trawlerman project, written off at a cost of £34 million [1] and a project for the Dept of Environment (Northern Ireland) where estimated costs were found to be doubled some two and a half years into the project [8]. Although investigations of such failed projects reveal a number of issues, such as problems with requirements and faulty software, problems associated with poor effort estimation, such as unrealistic budgets and timescales will almost invariably be found to have contributed to the problem.

Over the years a variety of techniques have been proposed to help solve the problem of making accurate, yet early, software project predictions. Many of these techniques involve the use of historical data in order to develop prediction systems. An example is statistical regression. Other

approaches involve the use of general models or prediction systems that are parameterised to account for differences between project environments. Examples include [4], and proprietary methods such as SLIM [13]. Often the disadvantage with such approaches is the need for quantifiable inputs which are not available at the early stages of a project, such as the bidding stage, where the accuracy of a estimate has implications for the rest of the project.

Unfortunately no prediction technique has proved consistently accurate, even when we relax the accuracy criterion to merely require that a technique generates *useful* predictions. Worse still some techniques have proved consistently unsuccessful. For this reason there has been growing interest in recent years in exploring a variety of machine learning (ML) techniques either as a complement or an alternative to existing techniques. Examples the use of artificial neural nets [24], rule induction [15] and case based reasoning or analogy [21]. Although, on occasions researchers report impressive results, what is often less visible is the amount of hidden effort to configure a ML system. Typically there is a large search space — since many decisions must be made — and little theory to guide the would-be predictor. Consequently, searching for an effective prediction system frequently degenerates into an exercise of trial and error.

The Empirical Software Engineering Research Group at Bournemouth have been involved in the development of case based reasoning (CBR) techniques and tools to build software effort prediction systems for five years. Over this period we have had some success in producing more accurate results than traditional regression based techniques [2, 21, 22]. Subsequently, other research groups have reported more mixed experiences. We believe there are a variety of reasons why this may be so. First, and foremost, effectiveness of any prediction technique depends upon characteristics of the dataset. For example, regression will do well if the majority of datapoints fall upon some hyperplane. Conversely, CBR may well be favoured when discontinuities exist in any underlying relationship between effort and other independent variables. However, we also believe there are a variety of decisions that must be made when utilising CBR techniques. Such decisions include feature subset selection, the number of analogies to utilise and choice of adaptation strategy. This paper is intended to provide some experimental data to assist with the more effective use of CBR techniques for building prediction systems. It also considers some of the issues involved in making comparisons between competing prediction systems.

The remainder of the paper is structured as follows. The next section describes CBR in more detail and discusses the various experiences of different research teams. Next we outline our experimental method and provide some background on the datasets. This is followed by an analysis of the relationship between number of analogies and cases. We conclude by making some tentative recommendations for other researchers using CBR.

2. Related Work

The idea of using analogies as a basis for estimating software project effort is not new. Boehm [4] suggested the informal use of analogies as a possible technique almost 20 years ago. The idea was reiterated by Cowderoy and Jenkins [7] in 1988 but again with no formal mechanism for selecting analogies. The next development was from Vicinanza *et al.* [17, 19, 23] who suggested that developments from the machine learning community in the form of CBR might be usefully adapted to help make better software project predictions.

The approach of Vicinanza *et al.* is to use domain specific similarity measures coupled with rule based adaptation. They found their technique outperforms COCOMO and FPs for Kemerer's [11] dataset augmented with an additional 7 projects. One disadvantage of this approach is that it is

very specific to the particular dataset since the rules and similarity measures are defined in terms of features available. It is unclear how this approach could be easily generalised.
Bisio and Malabocchia [3] report accuracy levels in the region of MMRE[1] = 40 to 50%, for CBR again using the COCOMO dataset, however, the system was only able to make predictions for 46 out 63 projects.

Our approach contrasts with that of Vicinanza *et al*. in that we were seeking to develop a more general means of building prediction systems. Our belief is that collecting historical data is sufficiently challenging as it stands without the additional requirement of having predetermined feature sets. We prefer to allow estimators the freedom to utilise those features that they believe best characterise their projects and are most appropriate to their environments. Consequently, we use Euclidean distance in *p*-dimensional feature space as a means of measuring similarity between cases. Categorical features are treated as either identical or completely dissimilar. We adopt a simple analogy adaptation strategy of the mean of the *k* nearest neighbours, where *k* is an integer such that $0 < k \leq n$. When k=1 then the technique is a simple nearest neighbour method. As *k* tends towards *n* so the prediction approach tends towards merely using the sample mean. The choice of *k* is determined by the estimator.

Our analogy based approach to project estimation is implemented in a software tool known as ANGEL. Using ANGEL we were able to compare the levels of accuracy we obtained with stepwise regression (SWR). We found that for all nine (independent) datasets we studied, ANGEL generated better results than SWR with the exception of one dataset where results were equally good (see Shepperd and Schofield [21]). Subsequently other research groups endeavoured to replicate these findings, with mixed results. Niessink and van Vliet [18] reported that CBR out performed SWR, and Finnie *et al*. [10] found that CBR outperformed regression analysis. By contrast, Briand *et al.* [5] and Stensrud and Myrtveit [25] reported that regression based analysis generated more accurate models than CBR. Why should this be the case? As we have already stated there is likely to be a strong interaction between the accuracy of a given prediction system and underlying characteristics of the dataset it is applied to. Briand *et al.* report very high adjusted R-squared values for their regression based prediction system. The obvious conclusion is that majority of datapoints fall close to a hyperplane. Such circumstances will favour regression since it will sensibly interpolate and extrapolate. Conversely, an analogy based approach tends to explore what *existing* datapoint, or clump of datapoints, is most similar to the target case.

The review of these results indicates CBR is a more complex technology involving more design decisions than we first thought. The problems that either we, or the other research groups, perceive are:

- feature subset selection
- scaling
- similarity measure
- how many analogies to use (i.e. finding a suitable value for *k*)
- analogy adaptation

In this paper we concentrate on the fourth problem, the number of analogies. For discussion of the other issues see [14]. ANGEL allows the estimator to choose the number of analogies. In our published results we used between one and three analogies. Briand *et al.* use a single analogy.

[1] MMRE or mean magnitude of relative error is due to Conte *et al.* [6] and is the mean absolute percentage error.

Which is "best"? How can we know? This area has not been systematically explored, in particular the interaction that one might expect between *k* and *n*.
To summarise this section, we note that different research groups have had varying experiences using CBR to predict software project effort. We posit that there are two explanations. The first is that underlying dataset characteristics will be influential in favouring or inhibiting different techniques for building prediction systems. CBR is no exception. Elsewhere we are conducting research into this topic [20]. The second explanation is that design decisions when building a CBR prediction system are also influential upon the results.

3. Method

The following analysis is based on a project effort data derived from two sources. The first data set has been collected from a Canadian software house [9]. We chose this dataset since it is publicly available, it is reasonably large by software engineering standards (81 cases of which 77 are complete) and combines both continuous and categorical features (8 continuous or discrete and 1 categorical). These factors were important since we required some flexibility for our experimentation into the interaction between different CBR strategies and the dataset characteristics. The only pre-processing was to discard the incomplete cases.

Data Set	Mean	Median	Min	Max	Skewness
Des77	4833.91	3542	546	23940	1.998

Table 1: Summary Statistics for the Dependent Variable (Effort)

Table 1 indicates that Des77 is positively skewed and is rather heterogeneous with an almost 20-fold variation in the dependent variable (effort).

In order to generate predictions we adopted a jack-knifing procedure for each dataset, as a means of avoiding very small validation sets. Consequently for each dataset there are *n* predictions, each based upon the remaining *n*-1 cases. For each prediction there is a corresponding residual, or error, being the difference between actual and predicted.

Data Set	Mean Absolute residual	MMRE
Des77	3008	119.3%

Table 2: Sample Mean Based Prediction Accuracy

In all our investigations we use prediction based upon the sample mean as our benchmark or null hypothesis. In situations where one is not predicting better than can be attained through the sample mean, it is questionable as to whether one is predicting at all! Table 2 indicates the threshold values for the four versions of the Desharnais dataset using two different accuracy indicators, namely mean absolute residual or error and MMRE. Our preference is for mean absolute error since it is a symmetric[2] measure, however, it is dataset dependent so we also provide the more conventional MMRE indicator to allow some comparability with other results. We would caution the reader though, that the two indicators could provide contradictory results on occasions. In such circumstances we consider mean absolute residual to be a better guide. A more detailed discussion of different prediction accuracy indicators can be found [12].

[2] By symmetric we mean an accuracy indicator that doesn't penalise under and over estimates differently.

Another problem is determining how much significance to attach to differences in accuracy indicators. Suppose prediction system A yields an MMRE of 50% and prediction system B yields an MMRE of 48%. What, if anything may we conclude from these observations? Our approach here is to test to see if the residuals are samples drawn from the same underlying population. Since we use absolute values the residuals are strongly positively skewed consequently we use non-parametric tests. Where the data is naturally paired, for example comparing k=1 and k=2 on the *same* dataset we use the Wilcoxon Signed Rank Test, otherwise the Mann-Whitney U Test. We set the significance level α=0.05. Since we are making multiple comparisons it could be argued that we should make the Bonferroni adjustment (i.e. adjusted significance level = α/x where x is the number of comparisons being made). This could be somewhat conservative, nevertheless we return to this in our discussion of results.

The second source of data was by means of simulation. One of the difficulties with naturally occurring data is that it can restrict systematic exploration of the factors affecting the performance of a CBR prediction system. Simulating data has been suggested by other researchers as being valuable, e.g. Michie *et al.* [16] point out that "in studying particular characteristics of algorithms, the role of simulations is crucial, as it enables controlled departures from assumptions, giving a measure of robustness etc.". (p127)

For this analysis we generated two data sets, one approximating to a Normal distribution (DS1) and the second exhibiting both extreme outliers (positively skewed) and multi-collinearity between the independent variables (DS2). The second, of course, is closer to most naturally occurring software project data sets. Each data set comprised four independent variables X1, … , X4 and two dependent variables Y1 and Y2. Y1 and Y2 were randomly generated by grammars based upon a range of operators and one or more of the independent variables. Y1 was generated from a continuous model whilst Y2 was discontinuous. This was done to simulate the heterogeneity that characterises many naturally occurring software engineering data sets.

With Des77 we had training sets of 76 cases — recall we used a jack knife — however, this relatively large for problem domain, so we used training sets of 20 cases and validation sets of 1000 cases. The significance of the latter will become apparent in later discussion.

The independent variables had error terms associated with them so they were supplied to the estimator as:

$$Y_1, Y_2, X'_1, \ldots, X'_n$$

where $X'_1 = X_1 + \varepsilon_1, \ldots, X'_n + \varepsilon_n$. The error terms were generated as normally distributed with a mean of zero and σ = 10% of the standard deviation of the sample of X_i. The simulation work was carried out blind such that the researcher building prediction systems was unaware of the true models for Y1 and Y2 or the true values for X1, … , X4. Further details may be found in [20].

4. Results

In this section we organise the results by data set and then by the following two questions:

- does the choice of the number of analogies k make any practical difference?
- if so, how can we *a priori* determine a value for k?

Clearly there are many decisions to consider when using CBR (for example how to measure similarity and feature subset selection to name but two), however,

4.1 Des77

(a) Does the choice of the number of analogies *k* make any practical difference?

We answered this question by performing a Kruskal-Wallis test[3] on the absolute residuals of all the predictions arising from *k*=1 to 5 (and also predicting using the sample mean so as to establish CBR adds some value). We found that there is some evidence that accuracy improves using more analogies, but we can not show statistical significance (p = 0.439).

(b) How can we *a priori* determine a value for *k*?

This part of our investigation is concerned with the relationship between the number of analogies (*k*) and the size of the dataset (*n*). We consider two approaches to determining *k*. First, it can be set to some constant value. We explore values in the range 1-5. Second, it can be determined dynamically as the number of cases that fall within distance *d*, of the target case. This has the effect that sometimes no prediction is possible since no analogies fall within the specified distance.

Accuracy Indicator	k=1	k=2	k=3	k=4	k=5	Sample Mean
MMRE	60.9%	55.3%	52.5%	47.6%	49.3%	119.3%
Mean abs residuals	2598	2417	2192	1979	1989	3008

Table 3: Accuracy by Number of Analogies for Des77

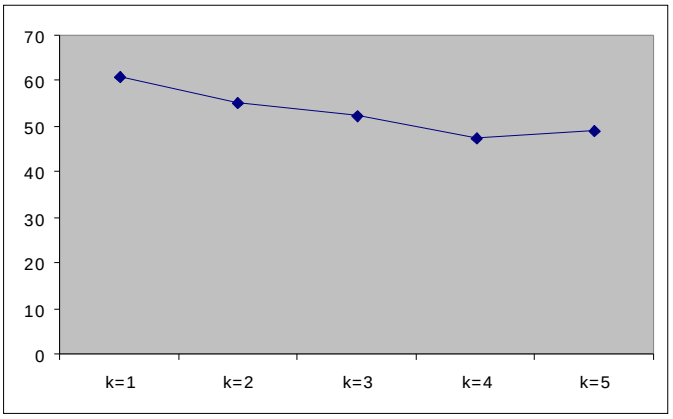

Figure 1: Lineplot of *k* vs MMRE

Figure 1 summarises Table 3 in a graphical form. We note some tendency in Figure 1 for accuracy to improve between *k*=1 and *k*=4 and thereafter to flatten out. Our analysis of the differences in accuracy has been somewhat subjective. The question remains as to how significant any of these differences are. As we indicated in the previous section we turn to the residuals to answer this question. For the purposes of this analysis we do not care about the

[3] This is a non-parametric version of ANOVA and is necessitated by the fact that we use absolute residuals.

direction of an error — an overestimate is deemed to be equivalent to an underestimate — consequently we use absolute residuals.

The next step is to assess whether any of the differences in accuracy shown by Table 2 and Figure 1 are significant. Using the Wilcoxon Signed Rank test we find even with the worst predictor (using k=1) that we still have a median error that is significantly lower than using the sample mean approach (p=0.0126). With a dataset of this size we are at last able to show some differences between the prediction systems. We are able to establish the following order of preference k=4, 5 < k=3 < k=1, k=2 where $x<y$ denotes x has the smaller mean absolute residual value. It would seem that a larger number of analogies tends to be more effective for prediction purposes.

4.2 Analysis Using Simulated Data Sets

We now turn to our simulated data sets DS1 (normal) and DS2 (skewed with multi-collinearity). Here we want to explore what impact some of the underlying characteristics of the data set have upon the relationship between k and accuracy.

(a) Does the choice of the number of analogies *k* make any practical difference?

As per the Des77 data set we answered this question by performing a Kruskal-Wallis test on the absolute residuals of all the predictions by data set and model.

Data set and model	p
DS1 - continuous model	0.000
DS1 - discontinuous model	0.000
DS2 - continuous model	0.341
DS2 - discontinuous model	0.000

Table 4: Significance Values for Kruskal-Wallis Tests

From Table 4 we see that in all cases, other than for DS2 and the continuous model, it does make a significant difference as to what value is selected for k. This means trying to better understand how to determine k a priori is an important research question. We are uncertain why DS2 and the continuous model is an exception although the explanation may lie in the complex nature of the data set and the relatively small size of the training set.

(b) How can we *a priori* determine a value for *k*?

Having determined that the value of k is important we now examine how two factors influence this relationship, namely (i) the type distribution of data set and (ii) the type of underlying model for the dependent variable.

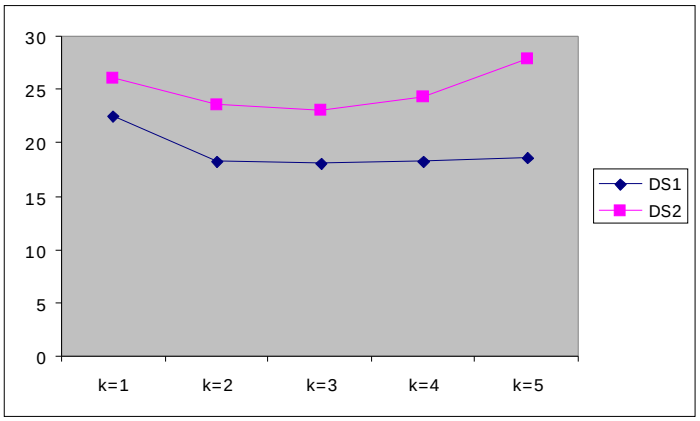

Figure 3: Lineplots of *k* vs MMRE for DS1 and DS2 for the Continuous Model

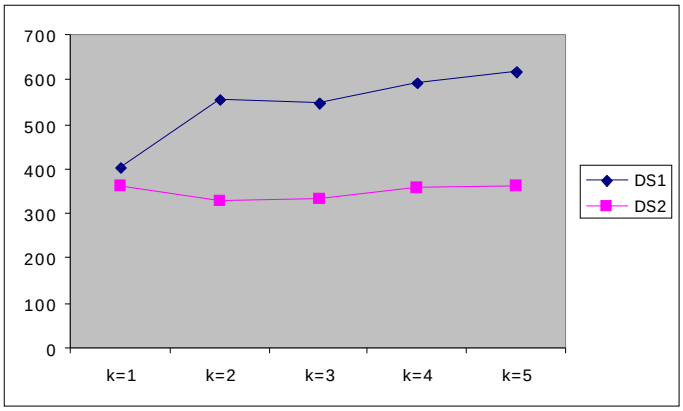

Figure 4: Lineplots of *k* vs MMRE for DS1 and DS2 for the Discontinuous Model

Comparing Figures 3 and 4 we see that, unsurprisingly, the predictions are substantially poorer when the underlying model is discontinuous in nature. For the continuous model we also see better predictions for DS1 where the data is normally distributed. This again is unsurprising. Less expectedly we see the reverse for the discontinuous model. Probably the only conclusion we can easily make is that both the model and data set DS2 have complex properties which interact in ways that are not always intuitive. This reinforces the need for further systematic exploration of this type of prediction since unfortunately it better represents the real world than the continuous model and normally distributed data of DS1.

In Figure 3 (with the continuous model) we see some evidence for *k*=2 being preferable to other values of *k* - p values are: .000, .044, .043, .000 when compared with 1, 3, 4 and 5 analogies respectively. Also in Figure 4 (the discontinuous model), k=1 is preferable – only significant when compared with values for 4 and 5 analogies with p values of .006 and .000 respectively. It is interesting to see that *k* is lower for the simulation studies than for Des77, almost certainly because the training set is much smaller. A more detailed investigation [14] found considerable evidence for such a relationship between k and n, the number of cases in the training set.

Unfortunately there are few other clear patterns. We believe this because we are studying complex phenomena and considerable further work is required to explore all the different

interactions between predictive performance, how to configure the CBR system and underlying characteristics of the data set.

5. Conclusion

In this paper we have discussed some of our experiences in investigating the problems that either other research groups or we perceive as hindering the effective use CBR technology for software project effort prediction. In particular, we have considered the impact of the choice of number of analogies when making predictions and whether we can *a priori* determine that choice. Our first source of data was the Desharnais dataset set that was collected from a Canadian software house. The second set of data were 4 simulated datasets, each with 20 cases as a training set and a 1000 cases as a validation set. The results show that using a larger number of analogies for a training set of 76 cases generated significantly better predictions. On the other hand, the nearest neighbour is observed to be the better predictor for a training set of 20 cases. We also found that the underlying distribution of the dataset does have a significant impact on the choice of the number of analogies. Moreover, predictions are substantially poorer when the underlying model is discontinuous in nature. It was difficult to identify patterns for datasets with complex distribution properties. We believe that a considerable amount of work is required to understand the relationship between the properties of the dataset and configuring a CBR system.

Acknowledgements
The authors are also grateful to Jean-Marc Desharnais for making his dataset available.

References

[1] *Computing*, 26/11/98
[2] Atkinson, K. and M.J. Shepperd. 'The use of function points to find cost analogies', in *Proc. European Software Cost Modelling Meeting* . Ivrea, Italy: 1994.
[3] Bisio, R. and F. Malabocchia. 'Cost estimation of software projects through case base reasoning', in *Proc. 1st Intl. Conf. on Case-Based Reasoning Research & Development* . Springer-Verlag, 1995.
[4] Boehm, B.W., *Software Engineering Economics* . Prentice-Hall: Englewood Cliffs, N.J., 1981.
[5] Briand, L., T. Langley, and I. Wieczorek. 'A replicated assessment and comparison of common software cost modeling techniques', in *Proc. ICSE (under review)* . 1999.
[6] Conte, S., H. Dunsmore, and V.Y. Shen, *Software Engineering Metrics and Models* . Benjamin Cummings: Menlo Park, CA, 1986.
[7] Cowderoy, A.J.C. and J.O. Jenkins. 'Cost estimation by analogy as a good management practice', in *Proc. Software Engineering 88* . Liverpool: IEE/BCS, 1988.
[8] Northern Ireland Audit Office: "The management of Information Technology in government departments in Northern Ireland", HMSO, Dd.750903.C8.2/92, 14567, Gp.118 from http://www.scit.wlv.ac.uk/~cm1995/cbr-tr/library.html
[9] Desharnais, J.M., *Analyse statistique de la productivitie des projets informatique a partie de la technique des point des fonction* . 1989, University of Montreal:
[10] Finnie, G.R., 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', *J. of Syst. Softw.* , 39, pp281-289, 1997.
[11] Kemerer, C.F. , 'An empirical validation of software cost estimation models', *CACM* , 30 (5), pp416-429 , 1987 .
[12] Kitchenham, B.A., *et al.* , 'Assessing Prediction Systems', *IEEE Transactions on Software Engineering (submitted)* , 1999.

[13] Putnam, L. H., ‘A General Empirical Solution to the Macro Sizing and Estimating Problem.’ IEEE Transactions on Software Engineering, 4(4), 345-361, 1978.

[14] Kadoda, G., Cartwright, M. H., Chen, L. and Shepperd, M. J., ‘Experiences Using Case-Based Reasoning to Predict Software Project Effort, ESERG Technical Report No. 00-09, Bournemouth University, 2000 (also published in the Procedings of EASE 2000)

[15] Mair, C., *et al.* , 'An investigation of machine learning based prediction systems', *J. of Syst. Softw.* , (accepted for publication), 1999.

[16] Michie, D., D.J. Spiegelhalter, and C.C. Taylor, ed. *Machine learning, neural and statistical classification* . Ellis Horwood Series in Artifical Intelligence, ed. J. Campbell. Ellis Horwood: Chichester, Sussex, UK, 1994.

[17] Mukhopadhyay, T., S.S. Vicinanza, and M.J. Prietula, 'Examining the feasibility of a case-based reasoning model for software effort estimation', *MIS Quarterly* , 16(June), pp155-71, 1992.

[18] Niessink, F. and H. van Vliet. 'Predicting maintenance effort with function points', in *Proc. Intl. Conf. on Softw. Maint.* Bari, Italy: IEEE Computer Society, 1997.

[19] Prietula, M.J., S.S. Vincinanza, and T. Mukhopadhyay, 'Software Effort Estimation With a Case-Based Reasoner', *J. Experimental & Theoretical Artificial Intelligence* , 8, pp341 - 363, 1996.

[20] Shepperd, M.J. and G. Kadoda, Comparing Techniques to Build Prediction System using Simulated Data. ESERG Technical Report No. 00/04, Bournemouth University, ESERG, 2000.

[21] Shepperd, M.J. and C. Schofield, 'Estimating software project effort using analogies', *IEEE Trans. on Softw. Eng.* , 23(11), pp736-743, 1997.

[22] Shepperd, M.J., C. Schofield, and B.A. Kitchenham. 'Effort estimation using analogy', in *Proc. 18th Intl. Conf. on Softw. Eng.* Berlin: IEEE Computer Press, 1996.

[23] Vicinanza, S., M.J. Prietula, and T. Mukhopadhyay. 'Case-based reasoning in effort estimation', in *Proc. 11th Intl. Conf. on Info. Syst.* 1990.

[24] Wittig, G. and G. Finnie, 'Estimating software development effort with connectionists models', *Information & Softw. Technol.* , 39(7), pp469-476, 1997.

[25] Myrtveit, I. and E. Stensrud, 'A controlled experiment to assess the benefits of estimating with analogy and regression models', *IEEE Trans. on Softw. Eng*., 25(4), pp510-525, 1999.