



An investigation of machine learning based prediction systems

Carolyn Mair^{*}, Gada Kadoda, Martin Lefley, Keith Phalp, Chris Schofield¹,
Martin Shepperd, Steve Webster

Empirical Software Engineering Research Group, Design, Engineering and Computing Department, Bournemouth University, P608, Poole House, Talbot Campus, Poole BH12 5BB, UK

Received 15 July 1999; received in revised form 2 November 1999; accepted 3 November 1999

Abstract

Traditionally, researchers have used either off-the-shelf models such as COCOMO, or developed local models using statistical techniques such as stepwise regression, to obtain software effort estimates. More recently, attention has turned to a variety of machine learning methods such as artificial neural networks (ANNs), case-based reasoning (CBR) and rule induction (RI). This paper outlines some comparative research into the use of these three machine learning methods to build software effort prediction systems. We briefly describe each method and then apply the techniques to a dataset of 81 software projects derived from a Canadian software house in the late 1980s. We compare the prediction systems in terms of three factors: accuracy, explanatory value and configurability. We show that ANN methods have superior accuracy and that RI methods are least accurate. However, this view is somewhat counteracted by problems with explanatory value and configurability. For example, we found that considerable effort was required to configure the ANN and that this compared very unfavourably with the other techniques, particularly CBR and least squares regression (LSR). We suggest that further work be carried out, both to further explore interaction between the end-user and the prediction system, and also to facilitate configuration, particularly of ANNs. © 2000 Elsevier Science Inc. All rights reserved.

Keywords: Machine learning; Neural net; Case-based reasoning; Rule induction; Software cost model; Software effort estimation; Prediction system

1. Background to research

Every day, businesses need to decide how to allocate valuable resources based on predictions. Unfortunately whilst most practitioners recognise the importance of accurate predictions of development effort for tendering bids, monitoring progress, scheduling resources and evaluating risk factors, current estimation techniques are often highly inaccurate. Traditionally, researchers have estimated software effort by means of off-the-shelf algorithmic models such as COCOMO (Boehm, 1981) where effort is expressed as a function of anticipated size; or have developed local models using statistical techniques such as stepwise regression (Kok et al., 1990). As algorithmic approaches are often unable to ade-

quately model the complex set of relationships that are evident in many software development environments, the results are frequently inaccurate. For example, Kemerer (1987) and Conte et al. (1986) frequently found errors considerably in excess of 100% even after model calibration. More recently, attention has turned to a variety of machine learning (ML) methods to predict software development effort. Artificial neural nets (ANNs), case-based reasoning (CBR) and rule induction (RI) are examples of such methods (see Karunanithi et al., 1992; Gray and MacDonell, 1997; Jorgensen, 1995). This paper outlines some comparative research into the use of ML methods to build software cost prediction systems.

2. Machine learning

ML techniques embody some of the facets of the human mind that allow us to solve hugely complex problems at speeds which outperform even the fastest computers (Schank, 1982). ML techniques have been

^{*} Corresponding author. Tel.: +44-1202-595560; fax: +44-1202-595314; web.: <http://dec.bmth.ac.uk/ESERG>.

E-mail addresses: cmair@bournemouth.ac.uk (C. Mair), gkadoda@bmth.ac.uk (G. Kadoda), mleffey@bmth.ac.uk (M. Lefley), kphalp@bmth.ac.uk (K. Phalp), mshelper@bmth.ac.uk (M. Shepperd), swebster@bmth.ac.uk (S. Webster).

¹ Chris Schofield is now with Nortel (cscho@nortelnetworks.com).

used successfully in solving many difficult problems such as speech recognition from text (Sejnowski and Rosenberg, 1987), adaptive control (Narendra and Parthasarathy, 1987; Hegazy and Moselhi, 1994) and markup estimation in the construction industry (Hegazy and Moselhi, 1994). Recently ML approaches have been proposed as an alternative way of predicting software effort.

This section describes three ML techniques that could be used in effort estimation: ANNs, CBR and RI. These techniques have been selected on the grounds that there exists adequate software tool support and because of their contrasting vantage points.

2.1. Artificial 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.

Fig. 1 shows an artificial neuron that computes the weighted sum of its n inputs, and generates an output of 1 if this sum is above a certain threshold u . Otherwise, an output of 0 results. Note that for back propagation algorithms a differentiable function, usually a sink, is used instead. Feed-forward multi-layer perceptrons are the most commonly used form of ANN, although many more sophisticated neural networks have been proposed. The ANN is initialised with random weights. The network then ‘learns’ the relationships implicit in a data set by adjusting the weightings when presented with a combination of inputs and outputs that are known as the training set. There are several training algorithms that can be used to train the network each having particular areas of speciality. Back propagation is the most common learning algorithm that has been used by software metrics researchers.

Most studies concerned with the use of ANNs to predict software development effort have focused on comparative accuracy with algorithmic models rather

than on the suitability of the approach for building software effort prediction systems. An example is the investigation by Wittig and Finnie (1997). They explore the use of a back propagation 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 validation sets are aggregated and yield a high level of accuracy (Desharnais MMRE = 27% and ASMA MMRE = 17%) although some outlier values are excluded. We note, however, that other factors such as explanatory value and configurability are equally important and also need investigation.

2.2. Case-based reasoning

CBR, originating in analogical reasoning, and dynamic memory and the role of previous situations in learning and problem solving (Schank, 1982), has received much attention recently. Cases are abstractions of events (solved or unsolved problems), limited in time and space. Aarmodt and Plaza (1994) describe CBR as being cyclic and composed of four stages:

1. *retrieval* of similar cases,
2. *reuse* of the retrieved cases to find a solution to the problem,
3. *revision* of the proposed solution if necessary, and
4. *retention* of the solution to form a new case.

When a new problem arises, a possible solution can be found by retrieving similar cases from the case repository. The solution may be revised based upon experience of reusing previous cases and the outcome retained to supplement the case repository. Consequently, issues concerning case characterisation (Rich and Knight, 1995), similarity (Aha, 1991; Watson and Marir, 1994; Koldoner 1993), and solution revision (Leake, 1996) must be addressed prior to CBR system deployment.

Examples of successful CBR tools for software project estimation include: Estor, a CBR system dedicated to the selection of similar software projects for the purpose of estimating effort, and more recently, FACE and ANGEL. A brief description follows.

Estor produces and adapts its own effort estimates using an analogy searching approach and rules inferred from the estimator’s own protocols. The performance of the estimates produced were comparable, in terms of R-squared values, to the expert’s own and far superior to those obtained using the regression based techniques, Function Points and COCOMO (see Vincinanza and Prietula, 1990; Srinivasan and Fisher, 1995).

Bisio and Malabocchia (1995) developed finding analogies for cost estimation (FACE), and assessed it using the COCOMO data set. In FACE all candidate

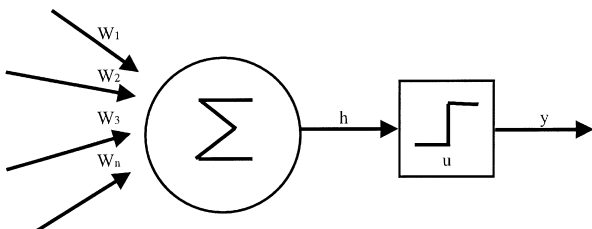


Fig. 1. A McCulloch and Pitts neuron.

analogies from the case repository were given a normalised score θ between 0 and 100 (100 being a perfect match) relating their similarity to the target case. The user could indicate the threshold (typically $\theta = 70$) over which cases can be used to form an estimate. If no cases were found (i.e. no cases have scores above the θ threshold score), then reliable estimation was not deemed possible. FACE appears to perform very favourably against algorithmic techniques.

ANGEL (Shepperd and Schofield, 1997) is another estimation tool based upon analogical reasoning. Here projects, or cases, are plotted in n -dimensional feature space and a modified nearest neighbour algorithm employed to identify the best analogies. Again they report results – derived from a number of datasets – of superior performance to LSR models.

Research shows that CBR systems can successfully be adapted to address the effort estimation problem. CBR approaches appear to have some advantages over other techniques, for example, effective functioning with small numbers of observations and in problem domains which are not well understood, and overt reasoning processes.

2.3. Rule induction

RI is a particular form of inductive learning in which algorithms produce rules as a result of modelling. RI is based on:

“... algorithms for induction which given a training set of examples, each of which is described by the values of an attribute and the outcome, will automatically build decision trees that will correctly classify not only all the examples in the training set, but unknown examples from the wider universe of examples of which the training set is presumed to provide a representative sample.” (Kennedy et al., 1997, p. 147).

Inductive learning is then the process of acquiring general concepts from specific examples. By analysing many examples, it may be possible to derive a general concept that defines the production conditions.

In order to produce a set of rules, induction works on a randomly, or algorithmically selected sub-set of the examples often referred to as the training set. These rules can be tested on the rest of the examples (the validation or test set) to assess how well they represent the data. RI can be used for a range of problems where there exists a set of suitable examples. Rules can be seen as decision trees where the leaf node contains the predicted value or range of values. Numeric decision trees are generated by calculating the average outcome for the set of cases being considered at each node. An example

fragment of rules generated from the Desharnais dataset is depicted below.

```
If AdjFPs ≥ 266 and
   If ExpPM < 3 and
     Transactions < 165
   Then effort = 3542
```

One advantage of inductive learning over neural network learning is that the rules are transparent and therefore can be read and understood. In the above example we see that adjusted function points is the first factor that is assessed followed by project manager experience then the number of transactions processed. Proponents of RI argue that this helps the estimator understand the prediction and any underlying assumptions upon which it is based. Moreover, the rules may be rephrased and provided to offer a clearer explanation as to how a prediction has been made. This is important in a problem domain such as project effort prediction since the estimator must trust the output, otherwise the prediction may be rejected. Trust is enhanced where some explanation for the output is available.

3. Method

In order to explore and compare the potential of the three machine learning techniques for building effort prediction models we selected an existing project effort dataset. The dataset comprised 81 software projects derived from a Canadian software house in the late 1980s (Desharnais, 1989). Despite the fact that this dataset is now 10 yr old, it is one of the larger, publicly available datasets. The approach we used, to partition the dataset into training sets and validation sets, is facilitated and enhanced with a greater number of data-points.

The dataset comprised 10 (one dependent and nine independent) features as summarised in Table 1. Four of the 81 projects contained missing values so were excluded from further investigation. The procedure adopted was to randomly partition the dataset into training sets of 67 projects and validation sets of 10

Table 1
Summary of the Desharnais dataset

Feature	Explanation
Project name	Numeric identifier
Effort	Measured in hours
ExpEquip	Team experience in years
ExpProjMan	Project manager's experience in years
Trans	Number of transactions processed
Entities	Number of entities
RawFPs	Unadjusted function points
AdjFPs	Adjusted function points
DevEnv	Development environment
YearFin	Year of completion

projects. This was performed three times yielding validation sets 1, 2 and 3 so as to help assess the stability of any prediction systems generated. In addition to ML techniques, we used a least squares regression (LSR) procedure to provide a benchmark comparison, again model fitting on the same training sets and testing on the remaining 10 projects. Accuracy was determined by mean magnitude of relative error (MMRE) statistic which provides an indication of the spread of estimation error. MMRE is also widely used in the literature so affords some comparability with other prediction systems.

The ANN work was based on a simple multi-layer perceptron with a back propagation learning algorithm using the software tool NeuFrame. In configuring the network we had to make design decisions concerning the topology, learning rate and momentum. Each configuration was also tested three times to assess the impact of different initial random weights for the nodes.

The CBR prediction system utilised the estimation by analogy tool ANGEL. In the past we have reported accuracy levels based upon a jackknifing procedure (Shepperd and Schofield, 1997), however, again for reasons of comparability, we used the training set to derive a regression model and the validation set to assess accuracy. We also utilised the facility within ANGEL to search for an optimal feature subset prior to using the validation set. Given that we only had nine features to contend with an exhaustive search was possible.

Finally, for the RI we used the data mining software package Clementine. Again we used the same three training and associated validation sets for this analysis. In addition, we carried out a preliminary investigation of feature subset selection but without automated support.

4. Results

Table 2 shows the accuracy levels (reported as MMRE) achieved for each technique using the three validation and associated training sets. It indicates considerable variation between the three validation sets. For example, RI ranges from MMRE = 41% to MMRE = 141%. Likewise the LSR ranges from 38% to 100%. This is disappointing and indicates all approaches are sensitive to changes in the training set and may not cope well with heterogeneity. The dataset contained a number of outlier values that contributed significantly to this problem.

In order to make comparisons between techniques, however, we provide the summary data in Table 3. We observe a ranking based on means would suggest that ANNs seem to be the most accurate technique. There is little to choose between CBR and LSR although there would seem to be greater variability with the latter

Table 2
Comparative accuracy of machine learning and LSR techniques

Method	Validation set	MMRE (%)
ANN	1	66
ANN	2	21
ANN	3	53
CBR	1	43
CBR	2	49
CBR	3	80
RI (subset selection)	1	41
RI (subset selection)	2	141
RI (subset selection)	3	89
RI	1	86
RI	2	140
RI	3	87
LSR – 1	1	47
LSR – 2	2	38
LSR – 3	3	100

Table 3
Summary statistics of prediction techniques

Technique	Count	MMRE			
		Mean	Median	Min	Max
ANN	3	47	53	21	66
CBR	3	57	49	43	80
LSR	3	62	47	38	100
RI	3	104	87	86	140
RI (subset selection)	3	90	89	41	141

which is clearly influenced by the outlier value for validation set 3. By comparison RI is consistently the least accurate technique.

Another observation is that there was a marked improvement – from 86% to 41% – for the RI method when applied to validation set 1 achieved by starting to explore feature subset optimisation. It would seem that the algorithm does not deal effectively with the categorical features indicating the type of development environment. When DevEnv is removed there is striking improvement in the accuracy of RI prediction system. Interestingly there is no similar improvement for validation sets 2 and 3 since the feature was only used deep in the tree and so only had to deal with a very small number of cases. As a consequence the removal of this feature had little impact upon the accuracy for the other validation sets. Nevertheless, this highlights an issue that RI based prediction systems also need to be configured prior to use.

Whilst on the surface it may appear that RI is the least accurate prediction technique it must be appreciated that the comparison is somewhat inexact. We have already noted that feature subset optimisation is a significant factor in achieving better levels of accuracy. ANGEL performs this search automatically reducing the feature set to {AdjFPs, DevEnv} for validation sets 1 and 2 and {Exp.Equip, AdjFPs, DevEnv} for

validation set 3. Parenthetically, we note that using the entire feature set has a significant negative impact, reducing average MMRE from 57% to 111% for the CBR technique. This suggests feature selection is potentially an important aspect of configuring a prediction system. The comparison, may be somewhat biased since very limited analysis of feature subsets was carried out for RI.

5. Discussion

This study has evaluated three machine learning techniques used to make software project effort predictions. These have been compared with LSR as a form of benchmark. We believe that in order to assess the practical utility of these techniques it is necessary to consider them within the context of their interaction with an end-user, for example project managers. Software effort prediction has a number of distinct characteristics compared to many other ML applications. First, training sets are comparatively small. Second, the predictions generally have a high degree of significance to the estimator. This has the consequence that interaction, or collaboration, between the prediction system and the estimator is of great importance. The value of this interaction has been shown for software effort prediction through empirical research that has indicated that end-users coupled with prediction systems can outperform either prediction systems or end-users alone (Stensrud and Myrteit, 1998).

Allowing the end-user to participate in the prediction process may lead to two beneficial effects. First, as noted above, it may enhance accuracy. It may be that users provide some kind of sanity check on the systems, whilst the system allows them to manipulate far more characteristics than would be possible manually. Second, it may increase confidence in the prediction. This consideration is also important in order to avoid the situation where end-users reject a prediction system. Whatever mechanisms are being utilised, it is clear that although accuracy is an important consideration, it is not sufficient to consider the accuracy of prediction systems in isolation. Hence, in assessing the utility of these techniques, we have considered three factors: accuracy, explanatory value and configurability. Accuracy has been the primary concern of researchers and clearly it is of considerable importance; a prediction system that fails to meet some minimum threshold of accuracy will not be acceptable. However, we believe that accuracy, by itself, is not a sufficient condition for acceptance.

5.1. Accuracy

In order to compare the accuracy of our predictive systems the same dataset has been used throughout. In

each case, the data was partitioned into a training set of 67 projects, and a validation set of 10 projects.

When considering accuracy a number of indicators could be used, for example, the sum of squares of the residuals, the percentage error, and the MMRE. In choosing to focus on the MMRE, we have decided that the potential spread of error is of most significance to software projects. Examination of the results shows that the ANN technique appears to perform best followed by CBR and LSR and lastly RI. Interestingly, our results for the ANNs are less good than those reported by Wittig and Finnie (1997), although this may be, in part, due to the impact of outlier projects in some of the validation sets. We also note the impact of feature subset selection, and again the potential for human involvement.

5.2. Explanatory value

One of the benefits of RI is that it makes explicit the rules that are being used by the prediction system. This, it is argued, can lead to insights about the data being used. However, the partitions or branches can sometimes appear to be rather arbitrary and reliance upon them as genuinely meaningful indicators may be unwise. In addition, our experience of RI methods suggests that they can be unstable predictors, and possibly less accurate than other techniques.

CBR, or estimation by analogy, also has potential explanatory value since projects are ordered by degree of similarity to the target project. Indeed, it is instructive that this technique demonstrates the effectiveness of user-involvement in performing better when the user is able to manipulate the data and modify predicted outputs. However, although this suggests an understanding of the data by the user, it is not clear to what extent this understanding is enhanced by use of the toolset.

The neural nets used within this study do not allow the user to see the rules being used by the prediction system. It is difficult to understand an ANN merely by studying the net topology and individual node weights. If a particular prediction is in some sense surprising to the end-user, it is harder to establish any rationale for the value generated. By comparison, both RI and CBR appear to offer an advantage in this respect. However, we note that it may be possible, in principle, to extract rules from ANNs, although this is beyond the scope of this paper.

5.3. Configurability

The third factor in comparing prediction systems is what we term configurability. In other words how much effort is required to build the prediction system in order to generate useful results. Regression analysis is a well established technique with good tool support. Even allowing for analysis of residuals and so forth, little effort

Table 4
Comparison of LSR, ANN, CBR and RI prediction systems

Technique	Accuracy	Explanatory value	Configurability
LSR	2 =	1 =	1 =
ANN	1	4	4
CBR	2 =	1 =	1 =
RI	4	1 =	3

needs to be expended in building a satisfactory regression model. Likewise the CBR needs relatively little effort since the tool we utilised, ANGEL, automates feature subset selection. By contrast, we found it took considerable effort to configure the neural net and it required a fair degree of expertise. Although various sets of heuristics have been published on this topic we found the process largely to be one of trial and error. For this reason, it is difficult to see how ANN techniques could be easily be used within the project estimation context by end-users. Lastly, whilst we found that whilst RI was not particularly onerous this was at the expense of feature subset analysis and consequently accuracy. An exhaustive search of all possible subsets would be quite time consuming and with larger feature sets impossible! Debuse and Rayward-Smith (1997) explore this issue further and discuss the application of simulated annealing algorithms to the problem of feature subset selection.

We summarise the relative merits and demerits of the techniques in Table 4. The numbers indicate ranking where 1 is best and 4 worst. The table illustrates that if one adopts a broader perspective than merely focusing upon accuracy, neural nets no longer become the obvious choice for building prediction systems.

6. Conclusions

In this paper we have compared three machine learning techniques with a LSR model for predicting software project effort. These techniques have been compared in terms of accuracy, explanatory value and configurability. Despite finding that there are differences in prediction accuracy levels, we argue that it may be other characteristics of these techniques that will have an equal, if not greater, impact upon their adoption. We note that the explanatory value of both estimation by analogy (case-based reasoning) and rule induction, gives them an advantage when considering their interaction with end-users. We also have found that problems of configuring neural nets tend to rather counteract their superior performance in terms of accuracy. This preliminary research has shown the need for further investigation, particularly in finding appropriate configuration heuristics for neural nets. Whilst some heuristics have been published (e.g. Walczak and Cerpa,

1999), we unfortunately did not find them to be of great value for this particular prediction task. In terms of prediction accuracy, the ML techniques we used are locally significant and are not generalisable. Nevertheless we believe that these ML methods warrant further investigation, particularly to explore under which conditions they are most likely to be effective.

Acknowledgements

This research was partly funded by grant (Grant GR/L37298) from the UK Engineering and Physical Sciences Research Council and the Defence Research Agency. The authors are also grateful to Jean-Marc Desharnais for making his dataset available.

References

- Aarmodt, A., Plaza, E., 1994. Case-based reasoning: foundational issues, methodical variations and system approaches. *AI Communications* 7 (1).
- Aha, W.D., 1991. Case-based learning algorithms. In: 1991 DARPA Case-Based Reasoning Workshop, Morgan Kaufmann, Los Altos, CA.
- Bisio, R., Malabocchia, F., 1995. Cost estimation of software projects through case based reasoning. In: International Conference on Case Based Reasoning, Sesimbra, Portugal.
- Boehm, B.W., 1981. *Software Engineering Economics*. Prentice-Hall, New York.
- Conte, S., Dunsmore, H.E., Shen, V.Y., 1986. *Software Engineering Metrics and Models*. Benjamin/Cummings, Menlo Park, CA.
- Debuse, J.C.W., Rayward-Smith, V.J., 1997. Feature subset selection within a simulated annealing data mining algorithm. *J. Intelligent Inform. Systems* 9, 57–81.
- Desharnais, J.M., 1989. Analyse statistique de la productivité des projets informatique a partie de la technique des point des fonction. Unpublished Masters Thesis, University of Montreal.
- Gray, A.R., MacDonell, S.G., 1997. A comparison of techniques for developing predictive models of software metrics. *Inform. Software Technol.* 39 (6), 425–437.
- Hegazy, T., Moselhi, O., 1994. Analogy based solution to markup estimation problem. *J. Comput. Civil Eng.* 8 (1), 72–87.
- Jorgensen, M., 1995. Experience with the accuracy of software maintenance task effort prediction models. *IEEE Trans. Software Eng.* 21 (8), 674–681.
- Karunanithi, N., Whitley, Malaiya, Y.K., 1992. Using neural networks in reliability prediction. *IEEE Software* 9 (4), 53–59.
- Kemerer, C.F., 1987. An empirical validation of cost estimation models. *Comm. ACM* 30 (5), 416–429.
- Kennedy, H.C., Chinniah, C., Bradbeer, P., Morss, L., 1997. The construction and evaluation of decision trees: a comparison of evolutionary and concept learning methods. In: Corne, D., Shapiro, J.L. (Eds.), *Evolutionary Computing*. AISB International Workshop, Manchester, UK, 7–8 April. Springer, LNCS 1305.
- Kok, P., Kitchenham, B.A., Kirakowski, J., 1990. The MERMAID approach to software cost estimation. In: *Proceedings of the Esprit Technical Week*.
- Leake, D., 1996. *Case-Based Reasoning: Experiences Lessons and Future Directions*. AAAI Press, Menlo Park.

- Narendra, K.S., Parthasarathy, K., 1987. Identification and control of dynamical systems using neural networks. *IEEE Trans. Neural Networks* 1, 4–27.
- Rich, E., Knight, K., 1995. *Artificial Intelligence*. McGraw-Hill, New York.
- Schank, R., 1982. *Dynamic Memory: A Theory of Reminding and Learning in Computers and People*. Cambridge University Press, Cambridge.
- Sejnowski, T.J., Rosenberg, C.R., 1987. Parallel networks that learn to pronounce english text. *Complex Systems* 1, 145–168.
- Shepperd, M.J., Schofield, C., 1997. Estimating software project effort using analogies. *IEEE Trans. Software Eng.* 23 (11), 736–743.
- Srinivasan, K., Fisher, D., 1995. Machine learning approaches to estimating software development effort. *IEEE Trans. Software Eng.* 21 (2), 126–136.
- Stensrud, E., Myrtveit, I., 1998. Human performance estimating with analogy and regression models: an empirical validation. In: *Proceedings of the Fifth International Metrics Symposium*, IEEE Computer Society, Bethesda, MD.
- Vincinanza, S., Prietula, M.J., 1990. Case based reasoning in software effort estimation. In: *Proceedings of the 11th International Conference on Information Systems*.
- Walczak, S., Cerpa, N., 1999. Heuristic principles for the design of artificial neural networks. *Inform. Software Technol.* 41 (2), 107–117.
- Watson, I., Marir, F., 1994. Case-based reasoning: a review. *The Knowledge Eng. Rev.* 9 (4), 327–354.
- Wittig, G., Finnie, G., 1997. Estimating software development effort with connectionist models. *Inform. Software Technol.* 39, 469–476.

Carolyn Mair is a research student at Bournemouth University. She received the M.Sc. in Research Methods at the University of Portsmouth in 1996, and the B.Sc. in Applied Psychology and Computing from Bournemouth University in 1995. Her research interests include biologically plausible artificial neural networks and vision.

Martin Shepperd is a Professor of Software Engineering at Bournemouth University. He received a Ph.D. in computer science from the Open University in 1991. He has published more than 60 refereed papers and three books in the field of empirical software engineering. He is also an editor of the journal *Information and Software Technology*. His research interests include software metrics and empirical software engineering.

Stephen Webster received the M.Sc. degree in Computer Science from Hertfordshire University (Hatfield Polytechnic) in 1987. Since 1987 he has worked at Bournemouth University. His research interests include object oriented analysis, design and programming methods and software process modelling. He is part of the programme committee for the Object Technology series of conferences. He is currently working for Semaphore Europe Ltd. as an OO consultant.

Gada Kadoda is a research Fellow at Bournemouth University. She received a Ph.D. in computer science from Loughborough University in 1997, the M.Sc. in Information Systems and Technology from the City University in 1993, and the B.Sc. in Computing, Mathematics and statistics from the University of Khartoum in 1991. Her research interests include formal methods, teaching tools and usability evaluation methods.

Keith Phalp graduated from the University of Kent with a degree in Mathematics in 1986. He completed a Masters degree in Software Engineering in 1991, and a Ph.D. in the area of Software Process Modelling, in 1994 (both at Bournemouth University). His research concentrates upon process modelling and its relationship to requirements engineering. He is a Chartered Mathematician and a member of the IMA.