Estimating Software Project Effort Using Analogies

Martin Shepperd and Chris Schofield

Abstract—Accurate project effort prediction is an important goal for the software engineering community. To date most work has focused upon building algorithmic models of effort, for example COCOMO. These can be calibrated to local environments. We describe an alternative approach to estimation based upon the use of analogies. The underlying principle is to characterize projects in terms of features (for example, the number of interfaces, the development method or the size of the functional requirements document). Completed projects are stored and then the problem becomes one of finding the most similar projects to the one for which a prediction is required. Similarity is defined as Euclidean distance in n-dimensional space where n is the number of project features. Each dimension is standardized so all dimensions have equal weight. The known effort values of the nearest neighbors to the new project are then used as the basis for the prediction. The process is automated using a PC-based tool known as ANGEL. The method is validated on nine different industrial datasets (a total of 275 projects) and in all cases analogy outperforms algorithmic models based upon stepwise regression. From this work we argue that estimation by analogy is a viable technique that, at the very least, can be used by project managers to complement current estimation techniques.

Index Terms—Effort prediction, estimation process, empirical investigation, analogy, case-based reasoning.

1 Introduction

An important aspect of any software development project is to know how much it will cost. In most cases the major cost factor is labor. For this reason estimating development effort is central to the management and control of a software project.

A fundamental question that needs to be asked of any estimation method is how accurate are the predictions. Accuracy is usually defined in terms of mean magnitude of relative error (MMRE) [6] which is the mean of absolute percentage errors:

$$\sum_{i=1}^{i=n}\left(\frac{\left|E-\hat{E}\right|}{E}\right)_i\frac{100}{n} \tag{1}$$

where there are n projects, E is the actual effort and $\hat{E}$ is the predicted effort. There has been some criticism of this measure, in particular that it is unbalanced and penalizes overestimates more than underestimates. For this reason Miyazaki et al. [19] propose a balanced mean magnitude of relative error measure as follows:

$$\sum_{i=1}^{i=n}\left(\frac{\left|E-\hat{E}\right|}{\min\left(E,\hat{E}\right)}\right)_i\frac{100}{n} \tag{2}$$

This approach has been criticized by Hughes [8], among others, as effectively being two distinct measures that should not be combined.

Other workers have used the adjusted R squared or coefficient of determination to indicate the percentage of variation in the dependent variable that can be "explained" in terms of the independent variables. Unfortunately, this is not always an adequate indicator of prediction quality where there are outlier or extreme values. Yet another approach is to use Pred(25) which is the percentage of predictions that fall within 25 percent of the actual value. Clearly the choice of accuracy measure to a large extent depends upon the objectives of those using the prediction system. For example, MMRE is fairly conservative with a bias against overestimates while Pred(25) will identify those prediction systems that are generally accurate but occasionally wildly inaccurate. In this paper we have decided to adopt MMRE and Pred(25) as prediction performance indicators since these are widely used, thereby rendering our results more comparable with those of other workers.

The remainder of this paper reviews work to date in the field of effort prediction (both algorithmic and non-algorithmic) before going on to describe an alternative approach to effort prediction based upon the use of analogy. Results from this approach are compared with traditional statistical methods using nine datasets. The paper then discusses the results of a sensitivity analysis of the analogy method. An estimation process is then presented. The paper concludes by discussing the strengths and limitations of analogy as a means of predicting software project effort.

2 A Brief History of Effort Prediction

Over the past two decades there has been considerable activity in the area of effort prediction with most approaches being typified as being algorithmic in nature. Well known examples include COCOMO [4] and function points[1] [2]. Whatever the exact niceties of the model, the general form tends to be:

• *M. Shepperd and C. Schofield are with the Department of Computing, Bournemouth University, Talbot Campus, Poole, BH12 5BB United Kingdom. E-mail: {mshepper, cschofie}@bournemouth.ac.uk.*

Manuscript received 10 Feb. 1997.
Recommended for acceptance by D.R. Jeffery.
For information on obtaining reprints of this article, please send e-mail to: tse@computer.org, and reference IEEECS Log Number 104091.

1. We include function points as an algorithmic method since they are dimensionless and therefore need to be calibrated in order to estimate effort.

$$E = aS^b \quad (3)$$

where E is effort, S is size typically measured as lines of code (LOC) or function points, a is a productivity parameter and b is an economies or diseconomies of scale parameter. COCOMO represents an approach that could be regarded as "off the shelf." Here the estimator hopes that the equations contained in the cost model adequately represent their development environment and that any variations can be satisfactorily accounted for in terms of cost drivers or parameters built into the model. For instance COCOMO has 15 such drivers. Unfortunately, there is considerable evidence that this "off the shelf" approach is not always very successful. Kemerer [12] reports average errors (in terms of the difference between predicted and actual project effort) of over 600 percent in his independent study of COCOMO. Other independent studies [14], [18] have also reported high error rates.

Another algorithmic approach is to calibrate a model by estimating values for the parameters (a and b in the case of (3)). However, the most straightforward method is to assume a linear model, that is set b to unity, and then use regression analysis to estimate the slope (parameter a) and possibly introduce an intercept so the model becomes:

$$E = a_1 + a_2 S \quad (4)$$

so that a_1 represents fixed development costs (for example regression testing will consume a fixed amount of effort irrespective of the size of the software) and a_2 represents productivity. Kok et al.[15] describes how this approach has been successfully utilized on the Esprit MERMAID Project.

Function points [2] are also often calibrated to local environments in order to convert size in function points to predicted effort. Again, as with COCOMO, quite mixed results have been reported [9], [10], [12], [17]. Kitchenham and Kansala [13] also note that better results can be obtained through disaggregating the components of function points and using stepwise regression to reestimate weights and determine the significant components.

Although, most research into project effort estimation has adopted an algorithmic approach there has been limited exploration of machine learning or nonalgorithmic methods. For example, Karunanithi et al. [11] report the use of neural nets for predicting software reliability, and conclude that both feed forward and Jordan networks with a cascade correlation learning algorithm, out perform traditional statistical models. More recently Wittig and Finnie [28] describe their use of back propagation learning algorithms on a multilayer perceptron in order to predict development effort. An overall error rate (MMRE) of 29 percent was obtained which compares favorably with other methods. However, it must be stressed that the datasets were large (81 and 136 projects, respectively) and that only a very small number of projects were withdrawn for validation purposes. Some outliers also appear to have been removed. This tends to confirm the findings from Serluca [25] that neural nets seem to require large training sets in order to give good predictions.

Another study by Samson et al. [24] uses an Albus multilayer perceptron in order to predict software effort. In this instance they use Boehm's COCOMO dataset. The work compares linear regression with a neural net approach using the COCOMO dataset. Both approaches seem to perform badly with MMREs of 520.7 and 428.1 percent, respectively.

Srinivasan and Fisher [27] also report on the use of a neural net with a back propagation learning algorithm. They found that the neural net outperformed other techniques and gave results of MMRE = 70 percent. However, it is unclear exactly how the dataset was divided up for training and validation purposes. Unfortunately, they also found that the results were sensitive to the number of hidden units and layers. Results to date suggest that accuracy is sensitive to decisions regarding the topology of the net, the number of learning epochs and the initial random weights of the neurons within the net. In addition, there is little explanation value in a neural net, that is such models do not help us understand software project development effort.

There have been a number of attempts to use regression and decision trees to predict aspects of software engineering. Srinivasan and Fisher [27] describe the use of a regression tree to predict effort using the Kemerer dataset [12]. They found that although it outperformed COCOMO and SLIM, the results were less good than using either a statistical model derived from function points or a neural net. Briand et al. [5] obtained rather better results (MMRE = 94 percent) from their tree induction analysis. In this case they used a combination of the Kemerer and COCOMO datasets. Porter and Selby [21], [22] describe the use of decision or classification trees in predicting aspects of the software development process. Results from this approach seem to be quite mixed and, as with the neural net approach, results are quite sensitive to aspects such as the choice of algorithm to derive the tree and tree depth.

Finally, Mukhopadhyay et al. [20] describe some early work using a hybrid case based reasoning (CBR) and rule based system. They report encouraging results based, again, upon the dataset collected by Kemerer, however, their approach requires access to an expert in order to derive estimation rules and create a case base. Our work differs in that no expert is used and a pure CBR strategy is adopted.

Although the results from nonalgorithmic approaches seem quite mixed they are sufficiently encouraging to warrant further investigation. However, we wish to stress that we do not propose that algorithmic approaches be rejected, merely that we search for additional and complementary methods of software project effort prediction. The reason for this is that in situations where pronounced linear or curvilinear relationships are to be found the ability to model this in terms of algorithms is important. In addition, the use of multiple techniques can be used as a "sanity check" upon any prediction generated.

3 Analogy

Estimation by analogy is a form of CBR. Cases are defined as abstractions of events that are limited in time and space. It is argued that estimation by analogy offers some distinct advantages.

- It avoids the problems associated both with knowledge elicitation and extracting and codifying the knowledge.
- Analogy-based systems only need deal with those problems that actually occur in practice, while generative (i.e., algorithmic) systems must handle all possible problems.
- Analogy-based systems can also handle failed cases (i.e., those cases for which an accurate prediction was not made). This is useful as it enables users to identify potentially high-risk situations.
- Analogy is able to deal with poorly understood domains (such as software projects) since solutions are based upon what has actually happened as opposed to chains of rules in the case of rule based systems.
- Users may be more willing to accept solutions from analogy based systems since they are derived from a form of reasoning more akin to human problem solving, as opposed to the somewhat arcane chains of rules or neural nets. This final advantage is particularly important if systems are to be not only deployed but also have reliance placed upon them.

The key activities for estimating by analogy are the identification of a problem as a new case, the retrieval of similar cases from a repository, the reuse of knowledge derived from previous cases and the suggestion of a solution for the new case. This solution may be revised in the light of actual events and the outcome retained to augment the repository of completed cases. This approach to prediction poses two problems. First, how do we characterize cases? Second, how do we retrieve similar cases, indeed how do we measure similarity?

Characterization of cases is largely a pragmatic issue of what information is available. Variables can be continuous (i.e., interval, ratio or absolute scale measures) or categorical (i.e., nominal or ordinal measures). When designing a new CBR system, experts should be consulted to try to establish those features of a case that are believed to be significant in determining similarity, or otherwise, of cases. Rich and Knight [23] describe the problem of choosing insufficiently general features. Again the solution appears to be to use an expert.

Assessing similarity is the other problem. There are a variety of approaches including a number of preference heuristics proposed by Kolodner [16]:

- *Nearest Neighbor Algorithms*. These are the most popular and are either based upon straightforward distance measures or the sum of squares of the differences for each variable. In either case each variable must be first standardized (so that it has an equal influence) and then weighted according to the degree of importance attached to the feature. A common algorithm is given by Aha [1].

$$SIM(C_1, C_2, P) = \frac{1}{\sqrt{\sum_{1 \in P} Feature\_dissimilarity(C_{1j}, C_{2j})}}$$

where P is the set of n features, C_1 and C_2 are cases and

$$Feature\_dissimilarity(C_{1j}, C_{2j}) \begin{cases} (C_{1j} - C_{2j})^2 \\ 0 \\ 1 \end{cases}$$

where 1) the features are numeric, 2) if the features are categorical and $C_{1j} = C_{2j}$, or 3) where the features are categorical and, $C_1 \neq C_{2j}$, respectively.

- *Manually guided induction*. Here an expert manually identifies key features, although this reduces some of the adavantages of using a CBR system in that an expert is required.
- *Template retrieval*. This functions in a similar fashion to query by example database interfaces, that is the user supplies values for ranges, and all cases that match are retrieved.
- *Goal directed preference*. Select cases that have the same goal as the current case.
- *Specificity preference*. Select cases that match features exactly over those that match generally.
- frequency preference—select cases that are most frequently retrieved.
- *Recency preference*. Choose recently matched cases over those that have not been matched for a period of time.
- *Fuzzy similarity*. Where concepts such as at-least-as-similar and just-noticeable-difference are employed.

The similarity measures suffer from a number of disadvantages. First, they tend to be computationally intensive, although Aha [1] has proposed a number of more efficient algorithms that are only marginally less accurate. However, efficiency is not an issue for project effort estimation as typically one is dealing with less than 100 cases. Second, the algorithms are intolerant of noise and of irrelevant features. One strategy to overcome this problem is to build in learning so that the algorithm learns the importance of the various features. Essentially, weights are increased for matching features for successful predictions and diminished for unsuccessful predictions. Third, symbolic or categorical features are problematic. Although there are several algorithms that have been proposed to accommodate such features they are all fairly crude in that they adopt a Boolean approach: features match or fail to match with no middle ground. A fourth criticism of these similarity measures is that they fail to take into account information which can be derived from the structure of the data, thus, they are weak for higher order feature relationships such as one might expect to see exhibited in legal systems.

Our approach has been guided by the twin aims of expediency and simplicity. In essence we take a new project, one for which we wish to predict effort, and attempt to find other similar completed projects. Since these projects are completed, development effort will be known and can be used as a basis for estimating effort for the new project. Similarity is defined in terms of project features, such as number of interfaces, development method, application domain and so forth. Clearly the features used will depend upon what data is available to characterize projects. The number of features is also flexible. We have analyzed data-

sets with as few as one feature and as many as 29 features. Features may be either categorical or continuous.

Similarity, defined as proximity in n-dimensional space (where each dimension corresponds to a different feature), is most intuitively appealing, hence we use unweighted Euclidean distance. The most similar projects will be closest to each other. Note that each dimension is standardized (between 0 and 1) so that it has the same degree of influence and the method is immune to the choice of units. Moreover, the notion of distance gives an indication of the degree of similarity. Once the analogous projects have been found, the known effort can be used in a variety of ways. We use the weighted or unweighted average of up to three analogies. No one approach is consistently more accurate so the decision requires a certain amount of experimentation on the part of the estimators. Because of the small datasets, we cope with noise (that is, unhelpful features that do not aid in the process of finding good analogies) by means of an exhaustive search of all possible subsets of the project features so as to obtain the optimum predictions for projects with known effort. The whole method, from storing analogies through eliminating redundant features to finding analogies is automated by a PC-based software tool known as ANGEL (ANaloGy Estimation tool[2]). A fuller description is to be found in Shepperd et al. [26].

4 Comparing Estimation by Analogy with Regression Models

Next, we compared the accuracy of software project effort prediction using analogy with an algorithmic approach based upon equations derived through stepwise regression analysis.

Table 1 summarizes the datasets that were used for our comparison of analogy based estimation with stepwise regression. As can be seen from the table the datasets are quite diverse and are drawn from many different application domains ranging from telecommunications to commercial information systems. All the data was taken from industrial projects, that is, no academic or student projects are included. The projects range in size from a few person months to over 1,000 person months. It is also important to stress that none of the data was collected with estimation by analogy in mind, instead we were able to exploit data that was already available. The final point is that we only utilized information that would be available at the time the prediction would be made, so we avoided project features such as LOC. This is important if we wish to avoid creating a false impression as to the efficacy of different prediction methods.

Table 2 shows the accuracy of the respective methods using the MMRE and Pred (25) values. A jack-knifing procedure[3] was adopted for the analogy-based predictions, since this could be automated in the ANGEL tool, the regression models were generated using the *entire* dataset. This means the results are likely to be biased in favor of the regression models. Note that we use two slightly different regression analysis techniques. Both regression 1 and 2 use stepwise regression, however, regression 1 restricts the procedure to the three variables most highly correlated with the dependent variable (i.e., effort). Not surprisingly the results are in general similar, however, occasional differences are due to the fact that the regression procedure attempts to minimize the sum of the squares of the residuals, whereas MMRE is based upon the mean of the sum of the unsquared residuals.

Each dataset is treated separately since each one has different project features available and therefore we are not able to merge all the data into a single all encompassing dataset. This is appropriate since it is unlikely that an organization would have access to such large volumes of data and there seems some merit in estimating using smaller, more homogenous datasets, a point we will return to.

From Table 2 we see that for all datasets the MMRE performance of estimating by analogy is better than that of the regression methods. This suggests that analogy is capable of yielding more accurate predictions, at least for these datasets. An interesting problem occurs for Real-time 1 dataset. Here it was not possible to develop an algorithmic model or to use regression analysis since the dataset comprises only categorical data, with the exception of actual project effort. Indeed the dataset was very sparse and was made up of only three distinguishing project features. Yet even in these highly unpropitious circumstances the analogy method was able to yield a predictive accuracy of 74 percent. This is indicative of the possibility of being able to use analogy based estimation at an extremely early stage of a project when other estimation techniques may not be possible for the reason that analogy does not require quantitative data. Similarly, an accuracy of 39 percent was obtained for the dataset Telecom 1 despite the fact that only a single distinguishing feature was available. Again, stepwise regression only achieves a result of MMRE = 86 percent by method 1 or 2.

The Pred(25) results from Table 2 are slightly more mixed. Recall that unlike MMRE, a higher score implies better predictive accuracy. Two datasets (Atkinson and Desharnais) yield a higher Pred(25) score for the regression model. In general, the results are closer than for the MMRE analysis. One explanation lies in the fact that the ANGEL tool explicitly tries to optimize the MMRE result so that it is not surprising that it performs best in terms of this indicator. A second explanation lies in the fact that MMRE and Pred(25) are assessing slightly different characteristics of a prediction system. MMRE is conservative and looks at the mean absolute percentage error whereas Pred(25) is optimistic and focuses upon the best predictions (i.e., those within 25 percent of actual) and ignores all other predictions. The choice of indicator to some extent depends upon the objectives of the user. Nevertheless, the overall picture suggests that estimation by analogy tends to be the more accurate prediction method.

2. The authors are happy to provide a simple version of ANGEL at no cost. The zip files may be downloaded from http://xanadu.bournemouth.ac.uk/ComputingResearch/ChrisSchofield/Angel/AngelPage.html

3. Jack knifing is a validation technique whereby each case is removed from the dataset and the remainder of the cases are used to predict the removed case. The case is then returned to the dataset and the next case removed. This procedure is repeated until all cases have been covered.

TABLE 1
DATASETS USED TO COMPARE EFFORT PREDICTION METHODS

Name	Source	n	Features	Description
Albrecht	[2]	24	5	IBM DP Services projects
Atkinson	[3]	21	12	Builds to a large telecommunications product at U.K. company X
Desharnais	[7]	77	9	Canadian software house—commercial projects
Finnish	Finnish Dataset: dataset made available to the ESPRIT Mermaid Project by the TIEKE organization	38	29	Data collected by the TIEKE organization from IS projects from nine different Finnish companies.
Kemerer	[12]	15	2	Large business applications
Mermaid	MM2 Dataset: Dataset made available to the ESPRIT Mermaid Project anonymously	28	17	New and enhancement projects
Real-time 1	not in public domain	21	3	Real-time projects at U.K. company Z
Telecom 1	Appendix A	18	1	Enhancements to a U.K. telecommunication product
Telecom 2	not in public domain	33	13	Telecommunication product at U.K. company Y

TABLE 2
RELATIVE ACCURACY LEVELS OF EFFORT ESTIMATION FOR ANALOGY AND REGRESSION

Dataset	Analogy (MMRE) (%)	Regression 1 (MMRE) (%)	Regression 2 (MMRE) (%)	Analogy (Pred 25) (%)	Regression 1 (Pred 25) (%)	Regression 2 (Pred 25) (%)
Albrecht	62	90	90	33	33	33
Atkinson	39	45	40	38	43	38
Desharnais	64	66	66	36	42	42
Finnish	41[4]	101	128	39	21	29
Kemerer	62	107	107	40	13	13
Mermaid	78	252	226	21	14	14
Real-time 1	74	N/A	N/A	23	N/A	N/A
Telecom 1	39	86	86	44	44	44
Telecom 2	37	142	72	51	27	42

TABLE 3
RELATIVE ACCURACY LEVELS OF HOMOGENIZED DATASETS

Dataset	Analogy (MMRE) (%)	Regression 1 (MMRE) (%)	Regression 2 (MMRE) (%)	Analogy (Pred 25) (%)	Regression 1 (Pred 25) (%)	Regression 2 (Pred 25) (%)
Desharnais 1	37	41	41	47	45	45
Desharnais 2	29	29	29	47	48	48
Desharnais 3	26	36	49	70	30	50
Mermaid E	53	62	62	39	27	27
Mermaid N	60	–	–	25	–	–

In general, the best results seem to be achieved where the data is drawn from many builds or enhancements to an existing system, for example the Atkinson, Telecom 1, and Telecom 2 datasets. The poorest results occur when the data is drawn from a wide range of projects from more than one organisation, such as the Mermaid dataset. This tendency appears to be true for both analogy and regression analysis.

Table 3 shows the results of dividing the Desharnais and Mermaid datasets into more homogenous subsets. The Desharnais dataset is divided on the basis of differing development environments. The Mermaid data is divided into enhancement (E) and new (N) projects. We observe that this division leads to enhanced accuracy for all estimation methods. Overall analogy has equal or superior performance to regression based prediction for seven out of eight comparisons, the only exception being the Desharnais 2 dataset which reveals fractionally superior performance for regression based prediction when using the Pred (25) indicator. The Mermaid N dataset is particularly interesting as it shows a dataset for which no statistically significant relationships could be found between any of the independent variables and effort hence no statistically significant regression equation can be derived. By contrast, the analogy method is able to produce an overall estimation accuracy of MMRE = 60 percent.

Finally, we note that the procedure to search for optimum subsets of features for predicting effort reduced the set of features for every dataset studied excepting, of course, Telecom 1 which only had a single feature in the first place. This procedure has a significant impact upon the levels of accuracy that we were able to obtain.

5 SENSITIVITY ANALYSIS

An important question to ask about any prediction method is how sensitive is it to any peculiar characteristics of the data and how will it behave over time. All the datasets we studied were historical in the sense that they described completed

4. In a previous paper [26] we reported an accuracy level of MMRE = 62 percent. The improvement is due to the use of additional project features with which to find analogies that were not utilized during our earlier work.

projects and we conducted the analysis after the event. This section explores the dynamic behavior of effort prediction by simulating the growth of a dataset over time. This enables us to answer questions such as how many data points do we need for estimation by analogy to be viable and how stable are the results (in other words, are the accuracy levels vulnerable to the addition of a single rogue project)?

Figs. 1 and 2 show the trends in estimation accuracy as the datasets grow. The Albrecht dataset (Fig. 1) was selected as an example of a dataset for which a comparatively low level of accuracy was achieved and in contrast the Telecom 2 dataset (Fig. 2) showed the highest level of accuracy. The procedure was to randomly number the projects from 1 to n (where n is the number of projects in the dataset). Projects are added to the dataset, one at a time, in the random number order. Thus, the dataset grows until all projects are added. The optimum subset of features was recalculated as each new project was added. This involved for each partial dataset (starting from two projects), jack knifing the dataset by holding out each project, one at a time, and using the remaining projects to predict effort. The average absolute prediction error for all projects contained in the partial dataset gives the MMRE of that partial dataset. This procedure was repeated three times for each dataset (hence, A1, A2, and A3 and T1, T2, and T3).

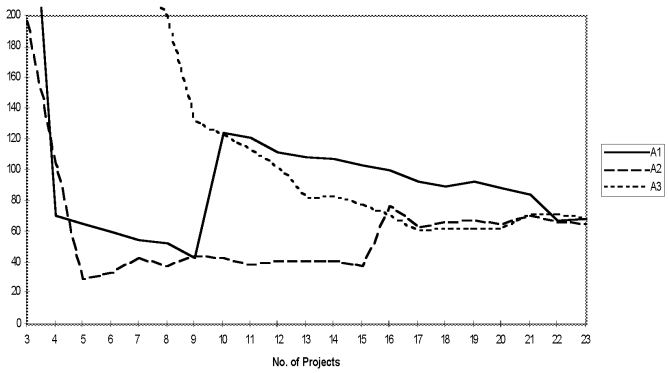

Fig. 1. Estimation accuracy over time (Albrecht dataset).

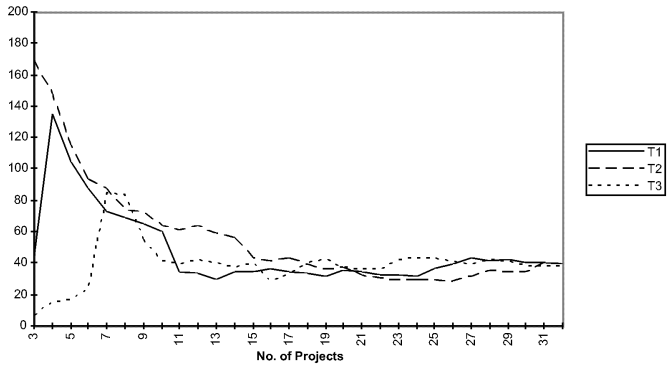

Fig. 2. Estimation accuracy over time (Telecom 2 dataset).

Overall, Figs. 1 and 2 show that the MMRE decreases as the size of the dataset grows. There is a tendency for the MMRE to start to stabilize at approximately 10 projects which suggests that estimation by analogy may be a high risk technique at below this number of projects. The Telecom 2 dataset shows little improvement beyond 15 projects. On this theme, it is interesting to note that, overall, it is not the largest datasets such as the Desharnais dataset that have the lowest MMREs and clearly other factors, over and above size, such as homogenity also have an impact.

An interesting feature of Fig. 1 is the sharp rise in the MMRE values that occurs after 10 projects have been added for random sequence A1 and 16 added for random sequence A2. Further investigation reveals that both of these anomalies are linked to the introduction of the same project. The project is third in sequence A3, when predictions are still very poor. This suggests that the results from estimating by analogy, like regression, can be influenced by outlying projects. However, A2 demonstrates that the affect of a rogue project is ameliorated as the size of the dataset increases. Superficially there appears to be a similar effect in Fig. 2 for sequences T1 and T3 and projects 4 and 7, respectively. In this case, however, the peaks are caused by different projects and the most likely explanation is the vulnerability of finding analogous cases from very small datasets.

6 An Estimation Process

This section considers how estimation by analogy can be introduced into software development organizations. The following are the main stages in setting up an estimation by analogy program:

- identify the data or features to collect
- agree data definitions and collection mechanisms
- populate the case base
- tune the estimation method
- estimate the effort for a new project

The first stage, that of identifying what data to collect, will be very dependent upon the nature of the projects for which estimates are required. Because of these variations, our software tool ANGEL is designed to be very flexible in the data that is used to characterize analogies and the user is able to define a template describing the data that will be supplied. Factors to be taken into account include beliefs as to what features significantly impact development effort (and are measurable at the time the estimate is required) and what features can easily be collected. There is little sense in identifying huge numbers of variables that cannot be easily or reliably collected in practice. Estimation by analogy can cope with both continuous and categorical data, although categorical data has to be held as binary values. For instance, programming language would be represented as a series of truth valued variables e.g., COBOL, 4GL, C++, etc. The reason for this is that the similarity measure treats categorical features as either being the same or different: there are no degrees of difference.

The second stage is to agree definitions as to what is being collected. Even within an organizations there may be no shared understanding of what is meant by effort. Any estimation program will be flawed, possibly fatally, if different projects are measuring the same features in different ways. It is also important to identify *who* is responsible for the data collection and *when* they should collect the data. Sometimes it

can be beneficial to have the same person collecting the data across projects in order to increase the level of consistency.

Next, the case base must be populated. Like all estimation methods, other than inspired guess work, analogy requires some data collection. Our experience suggests that a minimum of 10-12 projects are required in order to provide a stable basis for estimation. In general, more data is preferable although, in most cases, data collection will be an on-going process as projects are completed and their effort data becomes available. However, there appear to exist some tradeoffs between the size of the dataset and homogeneity. Again, our experience suggests there is merit in the strategy of dividing highly distinct projects into separate datasets. Often this separation is quite straightforward using such distinguishing features as application type or development site.

The penultimate stage is to tune the estimation method. The user also will need to experiment with the optimum number of analogies searched for, and whether to use a subset of variables, since some features may not usefully contribute to the process of finding effective analogies. Tuning can make quite a difference to the quality of predictions—typically tuning can yield a twofold improvement in performance—and for this reason the ANGEL tool provides automated support for this process.

The last stage is to estimate for a new project. It must be possible to characterise the project in terms of the variables that have been identified at the first stage of the estimation process. From these variables, ANGEL can be used to find similar projects and the user can make a subjective judgment as to the value of the analogies. Where they are believed to be trustworthy the prediction can be relied on to greater extent than where they are thought to be doubtful. Here we wish to sound a note of caution. The value of estimation by analogy as an *independent* source of prediction will be somewhat reduced if the users discount values that are not consistent with their prior beliefs and for this reason there was no expert intervention or manipulation in any of the foregoing analysis. Another indicator of likely prediction quality is the average MMRE figure obtained through jack knifing the dataset. Again a low figure will indicate more confidence than a high figure.

7 Conclusions

Accurate estimation of software project effort at an early stage in the development process is a significant challenge for the software engineering community. This paper has described a technique based upon the use of analogy sometimes referred to as case based reasoning. We have compared the use of analogy with prediction models based upon stepwise regression analysis for nine datasets, a total of 275 projects. A striking pattern emerges in that estimation by analogy produces a superior predictive performance in all cases when measured by MMRE and in seven out of nine cases for the Pred(25) indicator. Moreover, estimation by analogy is able to operate in circumstances where it is not possible to generate an algorithmic model, such as the dataset Real-time 1 where all the data was categorical in nature or the Mermaid N dataset where no statistically significant relationships could be found. We believe this type of situation may be quite common particularly at a very early stage in a project, for example in response to an invitation to tender. This makes analogy an attractive method for producing very early estimates.

Estimation by analogy also offers an advantage in that it is a very intuitive method. There is some evidence to suggest that practitioners use analogies when making estimates by means of informal methods [8]. Our approach allows users to assess the reasoning process behind a prediction by identifying the most analogous projects thereby increasing, or reducing, their confidence in the prediction.

Many experts have suggested that it is appropriate to use more than one method when predicting software development effort. We believe that estimation by analogy is a viable technique and can usefully contribute to this process. This is not to suggest that it is without weakness but on the empirical evidence presented in this paper it is certainly worthy of further consideration.

Appendix A

ACT	ACT_DEV	ACT_TST	CHNGS	FILES
305.22	250.49	54.73	218	105
330.29	225.4	104.89	357	237
333.96	177.35	156.61	136	98
150.4	114.7	35.7	25	24
544.61	357.49	187.12	263	197
117.87	71.5	46.37	39	39
1115.54	833.05	267.09	377	284
158.56	130.4	28.16	48	37
573.71	372.15	201.56	118	53
276.95	232.7	44.25	178	116
97.45	68.55	28.9	59	38
374.34	275.64	98.7	200	180
167.12	100.83	66.29	53	43
358.37	281.18	77.19	143	84
123.1	87.7	35.4	257	257
23.54	16.42	7.12	6	6
34.25	27.5	6.75	5	5
31.8	24.2	7.6	3	3

The above data is drawn from the dataset Telecom 1. ACT is actual effort, ACT_DEV and ACT_TEST are actual development and testing effort, respectively. CHNGS is the number of changes made as recorded by the configuration management system and files is the number of files changed by the particular enhancement project. Only FILES can be used for predictive purposes since none of the other information would be available at the time of making the prediction.

Acknowledgments

The authors are grateful to the Finish TIEKE organization for granting the authors' leave to use the Finnish dataset; to Barbara Kitchenham for supplying the Mermaid dataset; to Bob Hughes for supplying the dataset Telecom 2; and to anonymous staff for the provision of datasets Telecom 1 and Real-time 1. Many improvements have been suggested by Dan Diaper, Pat Dugard, Bob Hughes, Barbara Kitchenham, Steve MacDonell, Austen Rainer, and Bill Samson. This work has been supported by British Telecom, the U.K. Engineering and Physical Sciences Research Council under Grant GR/L37298, and the Defence Research Agency.

REFERENCES

[1] D.W. Aha, "Case-Based Learning Algorithms," *Proc. 1991 DARPA Case-Based Reasoning Workshop*. Morgan Kaufmann, 1991.

[2] A.J. Albrecht and J.R. Gaffney, "Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation," *IEEE Trans. Software Eng.*, vol. 9, no. 6, pp. 639-648, 1983.

[3] K. Atkinson and M.J. Shepperd, "The Use of Function Points to Find Cost Analogies," *Proc. European Software Cost Modelling Meeting*, Ivrea, Italy, 1994.

[4] B.W. Boehm, "Software Engineering Economics," *IEEE Trans. Software Eng.*, vol. 10, no. 1, pp. 4-21, 1984.

[5] L.C. Briand, V.R. Basili, and W.M. Thomas, "A Pattern Recognition Approach for Software Engineering Data Analysis," *IEEE Trans. Software Eng.*, vol. 18, no. 11, pp. 931-942, 1992.

[6] S. Conte, H. Dunsmore, and V.Y. Shen, *Software Engineering Metrics and Models*. Menlo Park, Calif.: Benjamin Cummings, 1986.

[7] J.M. Desharnais, "Analyse statistique de la productivitie des projets informatique a partie de la technique des point des fonction," masters thesis, Univ. of Montreal, 1989.

[8] R.T. Hughes, "Expert Judgement as an Estimating Method," *Information and Software Technology*, vol. 38, no. 2, pp. 67-75, 1996.

[9] D.R. Jeffery, G.C. Low, and M. Barnes, "A Comparison of Function Point Counting Techniques," *IEEE Trans. Software Eng.*, vol. 19, no. 5, pp. 529-532, 1993.

[10] R. Jeffery and J. Stathis, "Specification Based Software Sizing: An Empirical Investigation of Function Metrics," *Proc. NASA Goddard Software Eng. Workshop*. Greenbelt, Md., 1993.

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

[12] C.F. Kemerer, "An Empirical Validation of Software Cost Estimation Models," *Comm. ACM*, vol. 30, no. 5, pp. 416-429, 1987.

[13] B.A. Kitchenham and K. Kansala, "Inter-Item Correlations among Function Points," *Proc. First Int'l Symp. Software Metrics*, Baltimore, Md.: IEEE CS Press, 1993.

[14] B.A. Kitchenham and N.R. Taylor, "Software Cost Models," *ICL Technology J.*, vol. 4, no. 3, pp. 73-102, 1984.

[15] P. Kok, B.A. Kitchenham, and J. Kirakowski, "The MERMAID Approach to Software Cost Estimation," *Proc. ESPRIT Technical Week*, 1990.

[16] J.L. Kolodner, *Case-Based Reasoning*. Morgan Kaufmann, 1993.

[17] J.E. Matson, B.E. Barrett, and J.M. Mellichamp, "Software Development Cost Estimation Using Function Points," *IEEE Trans. Software Eng.*, vol. 20, no. 4, pp. 275-287, 1994.

[18] Y. Miyazaki and K. Mori, "COCOMO Evaluation and Tailoring," *Proc. Eighth Int'l Software. Eng. Conf.* London: IEEE CS Press, 1985.

[19] Y. Miyazaki et al., "Method to Estimate Parameter Values in Software Prediction Models," *Information and Software Technology*, vol. 33, no. 3, pp. 239-243, 1991.

[20] T. Mukhopadhyay, S.S. Vicinanza, and M.J. Prietula, "Examining the Feasibility of a Case-Based Reasoning Model for Software Effort Estimation," *MIS Quarterly*, vol. 16, pp. 155-171, June, 1992.

[21] A. Porter and R. Selby, "Empirically Guided Software Development Using Metric-Based Classification Trees," *IEEE Software*, no. 7, pp. 46-54, 1990.

[22] A. Porter and R. Selby, "Evaluating Techniques for Generating Metric-Based Classification Trees," *J. Systems Software*, vol. 12, pp. 209-218, 1990.

[23] E. Rich and K. Knight, *Artificial Intelligence*, second edition. McGraw-Hill, 1995.

[24] B. Samson, D. Ellison, and P. Dugard, "Software Cost Estimation Using an Albus Perceptron (CMAC)," *Information and Software Technology*, vol. 39, nos. 1/2, 1997.

[25] C. Serluca, "An Investigation into Software Effort Estimation Using a Back Propagation Neural Network," MSc dissertation, Bournemouth Univ., 1995.

[26] M.J. Shepperd, C. Schofield, and B.A. Kitchenham, "Effort Estimation Using Analogy," *Proc. 18th Int'l Conf. Software Eng.*, Berlin: IEEE CS Press, 1996.

[27] K. Srinivasan and D. Fisher, "Machine Learning Approaches to Estimating Development Effort," *IEEE Trans. Software Eng.*, vol. 21, no. 2, pp. 126-137, 1995.

[28] G.E. Wittig and G.R. Finnie, "Using Artificial Neural Networks and Function Points to Estimate 4GL Software Development effort," *Australian J. Information Systems*, vol. 1, no. 2, pp. 87-94, 1994.

Martin Shepperd received a BSc degree (honors) in economics from Exeter University, an MSc degree from Aston University, and the PhD degree from the Open University, the latter two in computer science. He has a chair in software engineering at Bournemouth University. Professor Shepperd has written three books and published more than 50 papers in the areas of software metrics and process modeling.

Chris Schofield received a BSc degree (honors) in software engineering management from Bournemouth University, where he is presently studying for his PhD. His research interests include software metrics and cost estimation.