Software Development Cost Estimation using Analogy: A Review

Jacky Keung
NICTA Ltd., Sydney, Australia
CSE/UNSW, Sydney, Australia
Jacky.Keung@nicta.com.au

Abstract

Software project managers require reliable methods for estimating software project costs, and it is especially important at the early stage of software cycle. Analogy for software cost estimation has been considered as a suitable alternative to regression-based estimation method, and empirical studies have shown that it can be used successfully in many circumstances.

It is important for project managers to understand the strengths and weaknesses of each useful software cost estimation method, more importantly when to use these methods and at which stage of the software development.

This paper provides a comprehensive overview of the background and the underlying theory of analogy for software cost estimation, published in major software engineering journals and conferences over the past 15 years.

Investigation on the dataset quality evaluation and its relevance to the target problem for analogy are further discussed, the result allows researchers and project managers to familiarize the underlying nature of the analogy-based approach.

1 Introduction

There are a number of competing software cost estimation methods available for software developers to predict effort required for software development, from the intuitive expert opinion methods to the more complex algorithmic modeling methods and the analogy-based methods. Previous empirical software cost estimation studies have attempted to determine which method is the best, however these studies have produced conflicting results. For example Shepperd and Schofield [31] claimed analogy provided better prediction accuracy. This was supported by Angelis and Stamelos [3] who found analogy-based systems were far superior to other methods and by the more recent work of Mendes and Mosley [24], and Mendes and Kitchenham [23] on a large heterogeneous data set. In contrast, Myrtvelt and Stensrud [27] replicated previous studies conducted by Shepperd et al. [31], but found analogy was not better than regression, and they also suggested that the results are sensitive to experimental design. Similarly, Briand and Eman [6] and Morasca [25] found analogy-based systems were less robust than other methods, particularly when dealing with heterogeneous data sets. Jeffery and Ruhe [12] also concluded stepwise regression outperformed analogy-based systems with the ISBSG dataset.

Over the past 10 years, researchers tried to explain why different research teams have reported widely different results by using analogy. Most recently, Mair and Shepperd [22] undertook a systematic review to investigate these contradictory results. They reviewed 20 primary studies comparing regression and analogy conducted during the past decade, and concluded that there was no clear indication that regression was better than analogy or vice versa. They concluded that the mixed results are due to the characteristic of the dataset and the individual data points [29]. The implication is that the resultant prediction is sensitive to the data quality of individual datasets. Shepperd and Kadoda [29] have also studied this issue using simulation and arrived the same conclusion.

As suggested by Mair and Shepperd [22], we agree researchers and practitioners should be focusing on when to use technique A rather B, as opposed to attempting to identify one single method as the best method. What is important to a project manager is which method is most appropriate in his/her specific circumstances as many of these methods are context-specific, and what information and data are immediately available.

Despite of its great significance in the software cost estimation research, estimating by analogy however remains largely unfamiliar by many industry software developers today, especially developers from small to medium organizations. This paper is intended for project managers and researchers wishing to gain in-depth knowledges of analogy-based methods. It provides a comprehensive overview for whom interested in learning analogy-based approaches, and a review of the underlying

1530-0803/09 $25.00 © 2009 IEEE
DOI 10.1109/ASWEC.2009.32

theory including issues discovered in its applications for whom experienced experts.

Section 2 presents an overview of the background and the history of analogy. Section 3 introduces the operation of analogy and its relationship to the case based reasoning framework. The application of analogy in software cost estimation is then discussed in section 4. This is followed by the introduction of a number of supporting tools for analogy-based software cost estimation in Section 5. We then discuss our experience and known issues in the application of analogy with recommendations. Section 7 concludes the paper.

2 Estimating by Analogy

The underlying reasoning behind analogy is similar to a basic human reasoning process used by almost every individual on a daily basis to solve problems based upon similar event(s) that happened in the past. Analogy is not a new reasoning paradigm as it has been extensively studied and discussed by philosophers and scientists for thousands of years. The role of analogy was important in early Greek thought. It was used as a method of suggesting or supporting explanation of particular natural phenomena, indeed in some cases the only means of providing empirical evidence to bear to obscure or intractable problems in that era, especially astronomy and meteorology, where direct experimentation was generally out of the question [38]. The great ancient Greek philosopher Aristotle successfully analyzed and demonstrated analogy as a method of inference in natural science, however neither he nor any later Greek logician made much progress towards eliciting the other important functions of analogy [21].

Today, analogy is either the cognitive process of transferring information from the source analogue to the target, or a linguistic expression corresponding to such a process, where the notion of conceptual metaphor in cognitive linguistics may be equivalent to that of analogy. In general, analogy refers to the relation between the source analogue and the target measured by their similarity. In Merriam-Webster's dictionary the word Analogy is defined as *"inference that if two or more things agree with one another in some respects they will probably agree in others"*. Note that the word "probably" in the above statement, that simply implies if two or more objects are similar with respect to some characteristics and their similarity can only be considered as "probably similar" with respect to other characteristics. Because their similarity is not definite, but can be measured by means of a probability measure on their similarity. Without such a measure, their similarity cannot be objectively defined.

Analogy is also an important artificial intelligence problem solving paradigm commonly used in a wide range of problem solving situations. It is an inference or an argument from a particular to another particular. In software engineering, analogy is fundamentally different from other major artificial intelligence approaches, it does not relying solely on general knowledge of a problem domain to making association along generalized relationships between descriptors and conclusions.

In software engineering, we use Analogy to utilize the specific knowledge of previously experienced, concrete problem situations or cases. A solution is derived by finding in similar case(s), and reusing it in a new problem situation. Aamodt and Plaza [1] attempted to describe another important difference is that analogy-based reasoning is an approach to incremental, sustained learning, because a new experience is retained each time a problem has been resolved, making it possible for immediately available for new problems.

The Analogy-based reasoning (also known as case-based reasoning or CBR) field has grown rapidly in the past decade at the time of writing, and this is evident from the increased number of research papers at major conferences, such as the International Conference on Case-Based Reasoning, and in several major artificial intelligence journals such as the Journal of Artificial Intelligence. Numerous commercial tools and applications have been developed to support analogy across a wide range of application domains.

Analogy-based reasoning is often used, however, especially in software effort estimation as a synonym for case-based reasoning (CBR), to describe the typical case-based approach where experience is retained for future reference. Generally, these two terms have been used interchangeably in the research of software cost estimation, but precisely there are slight differences between the two terms. As described by [1], the term "Analogy-based" is often used to characterize methods that solve new problems based on past cases from a different-domain, while typical case-based methods focus on indexing and matching strategies for single-domain cases. Research on analogy-based reasoning is therefore a subfield concerned with mechanisms for identification and utilization of cross-domain analogies [10].

The major focus of analogy-based study has been on the reuse of a past case, a pattern mapping problem, this is to finding a way to transfer the solution of an identified source analogous (from case-base) to the present problem (target case). The performance of analogy depends on the availability of a dataset. Inter-domain dataset application has often been the focus of research because within-company datasets of sufficient size are usually unavailable. In software engineering, a cross-company dataset such as ISBSG is often used for this purpose.

3 Analogy and the CBR Process

Analogy follows the general case-based reasoning (CBR) process. This section provides a general overview of this process.

Aamodt and Plaza [1] described a 4-stage general CBR cycle, which consisting of:

1. **RETRIEVE** the most similar cases or cases to the target problem
2. **RESUSE** the past information and solution to solve the new problem
3. **REVISE** the proposed solution and to better adapt the target problem
4. **RETAIN** the parts of current experience in the case-base for future problem solving

The following diagram (Figure 1) depicts the general cyclical CBR process, showing important steps and interactions in each stage of its application process.

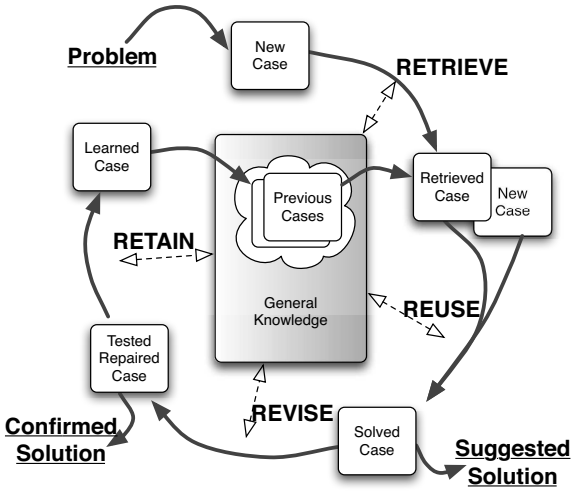

Figure 1. Anatomy of the CBR cycle

An initial problem description is a new case. This new case is used to RETRIEVE a case from the previous cases. The retrieved case is combined with the new case through REUSE into a solved case. Through the REVISE process this solution is tested for success. During RETAIN, useful experience is retained for future reuse, and the case based is updated by a new learned case, or by modification of some existing cases [1].

This 4-stage cyclical CBR process is sometimes referred to as the R^4 model [33]. When a new problem is entered into the CBR system, Shepperd [33] considered the new problem as a case that comprises two parts. There is a description part and a solution part forming the basic data structure of the system. The description part consists a vector of features that describe the case state at the point at which the problem is posed. The solution part describes the solution for the specific problem (the problem description part) .

The following diagram (Figure 2) illustrates the problem description part and the solution part forming the basic data structure of a typical CBR or analogy-based system.

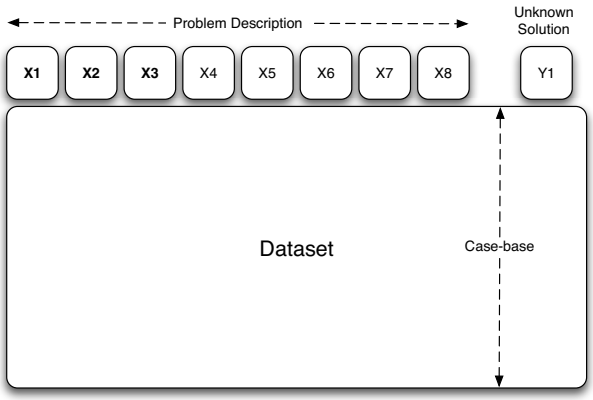

Figure 2. The basic data structure of a CBR system.

In Figure 2, for example, the effectiveness of similarity measures greatly depends on the overlap of features found in the case-base. Ideally the feature vectors from the target case and the case-base should be identical in configuration, since CBR does not easily deal with missing values [33].

There are various attempts to address the missing value problem in the existing literature, such as the imputation techniques described in [20]. In particularly, Cartwright et al. had explored the use of two simple data imputation techniques: Sample Mean Imputations (SMI) and k-Nearest Neighbor (k-NN) imputation for dealing with the problem of missing data. They found SMI to offer an improvement in data prediction accuracy over no imputation data, but that k-NN gave the best results. [7]

The similarity between the target case and each case in the case-base is determined by a similarity measure. Different methods of measuring similarity have been proposed for different measurement contexts. A similarity measure is measuring the closeness or the distance between two objects in a *n*-dimensional Euclidean space, the result is usually presented in a distance matrix (similarity matrix) identifying the similarity among all cases in the database. The Euclidean distance metric is probably the most commonly used in CBR for its distance measures. It is based on the principle of Pythagorean Theorem to derive a straight line distance between two points in n-dimensional space. The following diagram depicts the notion of Euclidean distance measure between two objects in 3-dimensional space (a, b, and c).

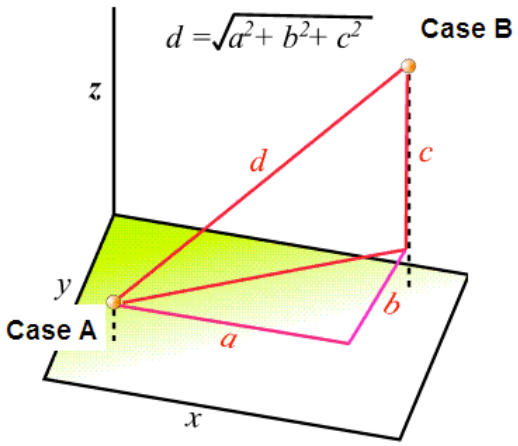

Figure 3. The Euclidean distance in 3-dimensional space.

In Figure 3, two objects are observed in the 3-dimensional space, in this instance, distance d between these two objects (case A and case B) are derived by the distance a, b and c using Pythagorean Theorem, the result is the distance between these two objects.

In general, the unweighted Euclidean distance between two points $P = (p_1, p_2, \ldots, p_n)$ and $Q = (q_1, q_2, \ldots, q_n)$, in Euclidean n-dimensional space, is defined and calculated as:

$$\sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2} \quad (1)$$

Another alternative is to apply weights to each feature to reflect the relative importance of each feature. The weighted Euclidean distance can be calculated as :

$$\sqrt{w_1(p_1 - q_1)^2 + w_2(p_2 - q_2)^2 + \cdots + w_3(p_n - q_n)^2} \quad (2)$$

where w_1 and w_n are the weights of 1^{st} and n^{th} project features. Keung and Kitchenham have discovered a dynamic solution to statistically determine a suitable weight w_i for each of the selected project features using statistical inference which improves the relevance of the dataset used as the case base [16] .

The Euclidean distance measure is suitable for general problems, particularly when values are of continuous nature. There are other different distance metrics for non-continuous variable, these include, but are not limited to Jaccard distance for binary distance [34] and Gower distance [9].

Irrespective of the similarity measure used in practice, the objective is to rank similar cases from the case base to the target case and utilize the known solution of the nearest k-cases. The value of k in this case has been the subject of debate [13, 31]. [31] suggested the ideal value for k is 3, that is, only three closest neighboring cases will be considered. These k cases will be adjusted or adapted to better fit the target problem by rules, a human expert or by a simple statistical procedure such as a simple average or a weighted mean. Once the actual value of the target case is available it can be reviewed and retained in the case-base for future reference. Stored cases must be maintained over time to prevent information irrelevancy and inconsistency. Keung [15] developed the Dynamic K-NN approach in an attempt to address the optimal K value in real time, provides vital information to construct an improved analogy-based model for effort prediction. The maintenance of an analogy-based model is a typical case of incremental learning in an organization utilizing the techniques of CBR.

4 The Application of Software Effort Estimation using Analogy

In software industry, estimates are usually produced by a domain expert with years of software development experience, in most of the case it is based on their own personal recollection of similar past events in the organization [19].

Although the expert-based approach is flexible and intuitive in a sense that it can be applied in a variety of circumstances where other algorithmic modeling and estimating techniques do not work, however there are no standard methods for expert opinion-based estimation.

Shepperd et al. [32] do not consider expert opinion an empirical method because the means of deriving an estimate are not explicit and therefore not repeatable, nor easily transferable to other staff. Knowledge relevancy is also problematic, as an expert may not be able to accurately identify estimates for a new application domain. It is also difficult for an expert to justify his/her assumptions in a way that permits such an estimate to be validated and accepted for decision making.

In contrast software estimation by analogy is a more formal and systematic approach to expert opinion using direct comparison with one or more past projects. The distinction between expert opinion and analogy in current software engineering research is that the former is a human-intensive approach and can based on variety of different methods such as rules of thumbs, personal recollection of past experiences etc. and the later is a data-intensive approach based on one or more specified potential analogous projects as discussed in the previous section, and can be automated and repeated [19].

A more complete and practical data-intensive analogy-based system was introduced in late 90s by Shepperd et al. who successfully demonstrated its potential for software effort prediction [31]. In many

circumstances, analogy provides an alternative to other data-intensive approaches, where sufficient data needed to fit a suitable model statistically is not available, but there is at least one analogous, or similar, project for which cost and schedule data is available.

The general application process for software effort estimation by analogy follows closely the CBR R^4 [1] model and involves the following number of steps:

1. Analogues case identification
The estimator reviews the new project, then measures or estimates the project feature metrics for the target project. The analogy-based system then searches the entire project case repository for the projects that are most similar to the target new project case and selects one or more similar projects as source analogues.

2. Target case prediction
The effort value of the source analogue(s) becomes an initial estimate for the target project. Various case adaptation techniques can be applied on the selected source analogue(s), such as simple average, weighted mean and linear regression on the effort values of the selected source analogues.

3. Final estimate adjustment
Final adjustment is applied on the initial effort and/or duration estimates in light of the differences between the target and source analogue(s) projects, and factors that are likely to influence effort and/or duration on the new project. This is usually based on expert judgement.

Once project effort is estimated, evaluated and reviewed, experience likely to be useful for future problem solving can be retained and stored back into the case base for future reference. However, full information about a target case is not available until the target project has been completed and its final effort and duration are known.

Each project case consists of a vector of project features, which may include, for example the number of input, output, and function points (Continuous), software development environment and programming language (Categorical). The feature vector can comprise different data types and this adds some complexity to the way in which distance between cases is measured. Euclidean distance metric is best suited to handling continuous data. A simple rectification is to transform categorical features into binary variables, i.e. dummy variables, and then normalize all other continuous features into the scale between 0 and 1, this allows features of both type equally comparable [31, 6]. The overall distance $dist(P_i, P_j)$ between two projects P_i and P_j is defined as:

$$dist(P_i, P_j) = \frac{\sqrt{\sum_{k=1}^{v} \delta(P_{ik}, P_{jk})}}{v} \qquad (3)$$

where v is the number of variables. The distance regarding a given variable k between two projects P_i and P_j is :

$$\delta(P_{ik}, P_{jk}) = \begin{cases} \frac{|P_{ik} - P_{jk}|}{max_k - min_k} & k \text{ is continuous} \\ 0 & P_{ik} = P_{jk} \\ 1 & P_{ik} \neq P_{jk} \end{cases} \qquad (4)$$

where value max_k and min_k are the maximum and minimum values of variable k. Equation 4 allows categorical and continuous project features to be combined, this allows distances between cases calculated correctly.

5 Analogy-based Tools and Systems

There are several implementations of analogy-based systems for software effort estimation. These systems followed the same principle of CBR reasoning approach explained above, their difference lies in their case adaptation and the number of source analogues. However their design goals are the same, which are automating the estimation process and enabling data-intensive analogy for software effort estimation. This section surveys related literature for typical analogy-based systems for software effort estimation, and briefly discusses their implementations.

5.1 ESTOR

ESTOR is an early implementation of an analogy-based tool to estimate software project effort. It was developed by [26] as a proof-of-concept system to evaluate the feasibility of case-based reasoning in software effort estimation. ESTOR uses function point components and inputs to the intermediate COCOMO model [5], it also assumes that estimators will use COCOMO project metrics. Similar to the CBR R^4 model discussed in previous section, ESTOR first selects an analogue for the target project by calculating the Euclidean distance between completed projects and selecting the nearest neighboring project. A set of rules are then applied to adjust the effort value for the analogue to account for the differences between the source and target project.

ESTOR considers two projects as its source of potential analogues. The projects were reconstructed using verbal protocols by an expert in analogical estimation. According to [35] and [36], this expert estimated effort accurately for a set of 10 projects. The adjustment rules used in ESTOR were derived from the same protocols [26]. The performance of ESTOR was assessed by means of absolute relative error or ARE for short. ARE can be expressed as:

$$ARE = 100 \times \frac{|(Actual - Estimated)|}{Actual} \qquad (5)$$

The mean ARE (MARE) can also be calculated for a set of estimates. This accuracy measure (also referred to as

Mean Magnitude Relative Error or MMRE for short) has been used widely in the software cost estimation literature, for example in [11, 14, 26].

Mukhopadhyay and Vincinanza [26] reported the performance for ESTOR for the expert's 10 estimates was 31%. When applied on the same 10 projects the (MARE) of ESTORs estimates was 51%. In another study, based on a 15 data-point industrial dataset, Kemerer [14] provided a comprehensive study to demonstrate ESTOR was comparable to the expert judgement approach and significantly more accurate than COCOMO model [5] and function points [2]. However as previously mentioned, the ESTOR approach requires access to an expert in order to derive rules for adaptation and to create a case base for reuse. Also Shepperd [33] points out that the rules are couched in terms of the particular set of features in Kemerers dataset which severely limits their applicability as there are wide discrepancies in the range and types of features collected by different software companies. Another similar tool, FACE [4] also using CBR technology, reported promising results based on the COCOMO dataset with accuracy level of MMRE = 40-50%. FACE suffers similar problems to ESTOR [33]. ESTOR was an early attempt at implementing an analogy-based system, it uses many of the same principles that we see in many modern analogy-based systems today, and provides significant contribution to the body of knowledge in domain of analogy for software effort estimation.

5.2 ACE

ACE Analogical and Algorithmic Cost Estimator, a prototype implementation of analogy-based system, has been developed by Walkerden and Jeffery [37] at the Centre for Advanced Empirical Software Engineering Research Group (CAESAR) in the late 90s. Similarly to ESTOR and other analogy-based system, ACE selects potential source analogues by means of a database of completed projects. ACE adjusts the actual effort required of the completed project to take account of the difference in size between the target and completed projects. ACE ranks all projects based on the differences between the target project and each completed project. The lower the rank the more similar the target project and the completed project. The project with the lowest mean rank is selected as the analogue for the target project.

If two completed projects differ from the target project by the same amount for a particular project feature, then they are allocated the same rank, and the rank of the project with the next lowest rank is adjusted accordingly. For example, if the target project has an adjusted function point (AFP) of 100, and two completed projects also happened to have an AFP of 100, then they are assigned rank 1. The next most similar project has an AFP of 95, and is then assigned rank 3. Categorical features are dealt equivalently. For example, projects with the same development environment are assigned rank 1, all other projects are then assigned the next rank and so on.

Once the accepted project with highest ranking have been identified, its effort value is then adjusted and adapted to estimate effort for the target project using linear extrapolation function based only on the dimension of a single size metric which is strongly correlated with effort, such as function point size counts. This linear size adjustment, based on unadjusted function points, is equivalent to using the productivity component of the analogue to predict the effort of the target project [37]. The performance of ACE is also measured in MARE. The linear extrapolation function is expressed as:

$$\text{Effort}_{TARGET} = \frac{\text{Effort}_{ANALOGUE}}{\text{FP}_{ANALOGUE}} \times \text{FP}_{TARGET} \quad (6)$$

For example, if the size measure for a target project is 320 function points, and a source analogue was identified with 350 function points, the effort required to complete the source analogue was 1,200 person-hour, then the effort estimate for the target project is estimated 1,097 person-hour using the linear extrapolation equation above.

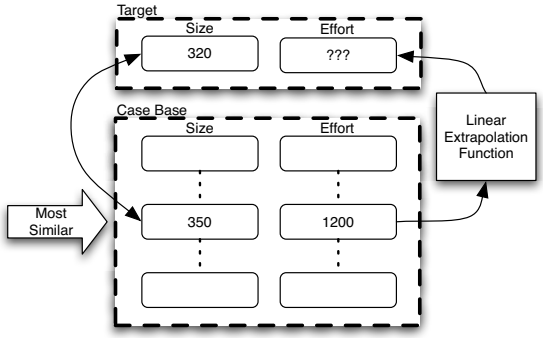

Figure 4. ACE linear extrapolation

Figure 4 demonstrates ACEs application of linear extrapolation function

The penetration of ACE, ESTOR and FACE in the software industry are very limited compared with the ANGEL system developed by Shepperd et al. [32].

5.3 ANGEL

archANGEL or ANGEL for short, is the dominant automated software effort estimation using analogy tool. It is widely known to both industry practitioners and researchers. Many research findings on analogy-based research published are based on the result produced by ANGEL, for example [30, 29, 31, 32, 33].

The ANGEL system is an important implementation of the case-based learning algorithm discussed in the previous section based on equation 3 and 4 where it adopts its similarity function and the normalization strategy for different data types of feature values.

ANGEL is popular for the completeness of its implementation. It supports various kinds of validation techniques, search heuristic algorithms to perform feature and case selection on the dataset, and a user-friendly graphical GUI as its primary user-interface. ANGEL does not assume that estimators will use a particular project dataset. The estimator can use whatever project dataset is available to setup an estimation instance in ANGEL. Similarity between target project and all potential source analogues from the case base are measured by the Euclidean distance metric.

The table below demonstrates the prediction power of analogy for the application of software cost estimation. The results were derived from an empirical evaluation of ANGEL based upon 9 real software project datasets [31].

Dataset	No. of Cases	No. of Variables	ANGEL (MMRE)	Stepwise Regression (MMRE)
Albrecht	24	5	0.62	0.90
Atkinson	21	12	0.39	0.45
Desharnais	77	9	0.64	0.66
Finnish	38	29	0.41	1.01
Kemerer	15	2	0.62	1.07
Mermaid	28	17	0.78	2.52
Real-time 1	21	3	0.74	N/A
Telecom 1	18	1	0.39	0.86
Telecom 2	33	13	0.37	1.42

Table 1. Relative prediction accuracy levels of using analogy and regression

The table above demonstrates ANGEL's prediction power in contrast with stepwise regression measured using MMRE, in this case, ANGEL significantly outperformed the regression approach.

ANGEL allows estimator specified project features to be used when ANGEL searches for source analogues. The best feature subset can also be determined by its built-in search algorithms by considers all possible combinations of feature subsets and selects the subset that minimizes the prediction performance metric, such as MMRE for the dataset, calculated by jack-knifing validation approach. It is important to note that the feature subset selection in ANGEL depends heavily on the performance metric selected (in most cases its accuracy is defined in terms of MMRE). MMRE (Mean Magnitude Relative Error) is one of the most common prediction accuracy measures used in software effort estimation research, it is formulated as [31]:

$$\sum_{i=1}^{n} \left(\frac{|E - \hat{E}|}{E} \right)_i \times \frac{100}{n} \tag{7}$$

where E is the actual effort and $\hat{E}$ is the predicted effort, given that there are n project cases to evaluate. There have been many criticisms of this measure for its unbalanced measure and penalises overestimates more than underestimates, for example described by Foss et al. [8]. Various other accuracy measures have been suggested such as the balanced mean magnitude relative error measure, but they too have limitations [8].

The accuracy metrics available in ANGEL are used to determine the best feature subset using brute-force on all possible combinations or heuristic search algorithms such as hill climbing and greedy search. These searching algorithms are computationally expensive, requiring a large amount of computing resources and time to complete, and it is not feasible when they are used with a large number of features in the dataset. In an early version of ANGEL, the maximum number of features in the dataset is limited to 10 [28]. An alternative is to use forward selection search algorithm reported by [18] for its improved efficiency and also yield very good prediction results. Even with successful completion of a search, the feature subset identified can not be justified with any statistical evidence. The only indication is the value of MMRE or any other accuracy measure used in the feature searching process.

ANGEL's feature selection process exists in its model preparation stage as shown in Figure 5. As we can see, the feature selection process in ANGEL will reduce the dimensionality of the dataset without producing any formal justification. However dimensionality reduction will influence its prediction outcome for the new target project.

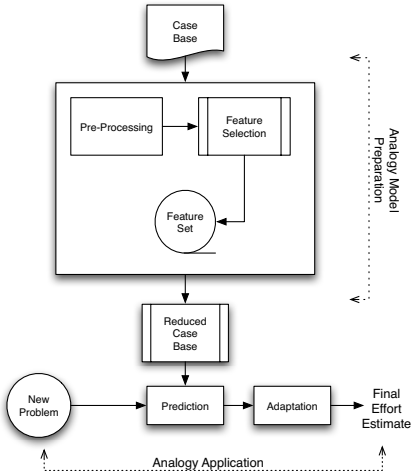

Figure 5. ANGEL application process

Figure 6 illustrates the work flow of a typical brute-force feature selection algorithm used in ANGEL to exhaustively evaluate all possible combinations of input features using jack-knifing, MMRE is used as the evaluation criteria for its prediction accuracy, the system works in a trial-and-error fashion.

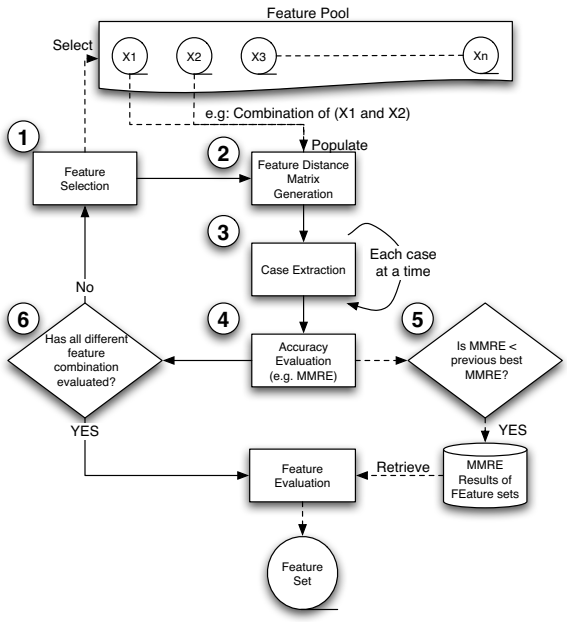

Figure 6. Brute-force feature subset selection process.

6 Discussion - Weaknesses in Analogy-Based Estimation

Despite the many benefits of analogy and the fact that the concept of estimating by analogy is relatively straightforward and that in many cases its performance is comparable or even better than most of the algorithmic models, there are drawbacks and difficulties with analogy-based estimation that temper its advantages and must be addressed.

Aha (1991) described some general deficiencies of the CBR algorithm, which is used in analogy-based systems such as ANGEL. One of the major issues at that time was they are computationally expensive because they save and compute similarities to all training cases, moreover:

- They are intolerant of noise.
- They are intolerant of irrelevant features.
- They are sensitive to the choice of the algorithm's similarity function.
- There is no simple way they can process categorical feature values.
- They give little usable information regarding the structure of the data.

These issues remain the same in case-based learning algorithm for more than 20 years, and still, few investigations have focused on their empirical limitations although algorithmic efficiencies have improved using various methods. For example, Shepperd et al. [31] were aware of that computational overhead of the case-based learning algorithm, but efficiency was not considered an issue for project effort estimation as software dataset are usually less than 100 cases and have limited number of features per case (less than 15).

Regardless of the computation resources required, the search algorithms used in the analogy-based systems will potentially influence the prediction outcome. Walkerden and Jeffery [37] described four important factors that influence prediction accuracy, they are:

1 The availability of an appropriate analogue (Dataset Relevancy)
2 The soundness of the strategy for selecting it (Feature Subset Selection)
3 The differences between the analogue and targets. (Distance Measures)
4 The accuracy of the data used. (Quality of the Dataset)

If there is no true analogue within a dataset for the target project, an analogy-based system may be continued to execute, and an analogue will be selected and used regardless of its appropriateness. For example, a completely unrelated project case may be selected as a potential source analogue because it appears similar (or nearest in feature metrics) to the target project. Walkerden

[37] noted that it is not clear how best to judge the appropriateness of a potential analogue for a target project. The current systems such as ESTOR and ANGEL use Euclidean distance between past projects and the target project to rank potential analogues, their usefulness as the basis of predictions depends on performance measures such as MMRE or MARE after a complex trial-and-error heuristic evaluation on the entire dataset.

Using ANGEL as a typical example, the current issue with software estimation are that the dataset being used is not evaluated statistically, it has a built-in evaluation engine using heuristic search algorithms such as hill climbing or brute-force search to compute the best combination of features, and project cases are based on the evaluation criteria MMRE or other variants to select the best feature subset. As such, the system will generate an estimate regardless of its dataset quality, or the availability of an appropriate analogue. In contrast, the appropriateness of a dataset in linear regression can be easily evaluated by the prediction power of the model, as measured by the multiple correlation coefficients and the probability value associated. Regression model also have mechanisms to identify extremely outlying cases. Using a statistical assessment of the suitability of the dataset means we assume the new target project comes from the same source as the other project. For analogy (CBR) a relevancy measure should be immediately available to evaluate the appropriateness of a dataset at its data preprocessing stage. This measure would help estimator to better understand the nature of the dataset used, and to decide whether an analogy-based approach is suitable for the target project under investigation, or an alternative approach should be sought according to the information available.

Analogy has the potential to mitigate problems with outliers, since estimating by analogy does not rely on calibrating a single model to suit all projects [37]. However, outliers in the dataset will have influence on the estimate. For example, if the target project is itself an outlier, it means that the dataset is inappropriate for the target problem. This effect may not be apparent to the estimator, as the current analogy-based systems have no mechanism in identifying whether a dataset is indeed appropriate for the purpose of estimating.

At the moment, there is a solution which aids estimator to exclude inappropriate analogues in the dataset. Keung et al. [17] developed the Analogy-X approach to overcome these issues using Mantel Statistics, which provides a statistical basis for analogy. More importantly it is able to detect a statistically significant predictive relationship and reject non-significant predictive relationship. Based on the same method it is also able to use a new project feature selection approach similar to that of stepwise regression commonly used in statistics. It has been reported in [17] that Analogy-X improved prediction performance and provides relevant solution to the problem under investigation.

Using analogy would be ideal in a situation where relevant and completed project data consistently measured are available. However the dynamic nature of software projects due to technological and organizational changes may eliminate the justification of using analogy at project level. Further research is required in this aspect leading to successful adoption of analogy in an organization.

7. Conclusion

Accurate estimates for software development continue to be a dream for many project managers. More importantly, managers must understand the nature of software development in their own context, and must understand the science of software engineering with respect to their business goal, and managerial strategies. This information cannot be easily captured using any particular modeling technique and tools. No software tool can replace the talents of a human expert, who is able to consider wider range of issues that lead to the success and failure of a software project. Software estimation techniques and tools can only be used to facilitate a human expert to identify some of these issues and provide recommendations. In this article, we provided a comprehensive overview of analogy-based software cost estimation and an up-to-date review of its latest development to allow project managers to better understand the underlying technologies involved and when to use analogy-based systems appropriately.

8. Acknowledgements

NICTA is funded through the Australian Government's Backing Australia's Ability Initiative, in part through the Australian Research Council.

References

[1] A. Aamodt and E. Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *Artificial Intellegence Communications*, 7:39–59, 1994.

[2] A. J. Albrecht and J. R. Gaffney. Software function, source lines of code, and development effort prediction: a software science validation. *IEEE Transactions on Software Engineering*, 9:639–648, 1983.

[3] L. Angelis and I. Stamelos. Reply to comments by m. jorgensen, on the paper: A similation tool for efficient analogy based cost estimation by l. angelis and i. stamelos. *Empirical Software Engineering*, 7(4), 2002.

[4] R. Bisio and F. Malabocchia. Cost estimation of software projects through case base reasoning. In *1st Intl. Conf. on Case-Based Reasoning: Research and Development*, 1995.

[5] B. Boehm. *Software Engineering Economics*. Prentice-Hall, NY, 1981.

[6] L. C. Briand, K. E. Eman, and K. D. Maxwell. An assessment and comparison of common software cost estimation modeling techniques. In *International Conference on Software Engineering*, LA, CA, 1999.

[7] M. H. Cartwright, M. J. Shepperd, and Q. Song. Dealing with missing software project data. In *Ninth International Software Metrics Symposium*, pages 154–165, 2003.

[8] T. Foss, E. Stensrud, B. Kitchenham, and I. Myrtveit. A simulation study of the model evaluation criterion mmre. *Software Engineering, IEEE Transactions on*, 29(11):985–995, 2003. 0098-5589.

[9] J. C. Gower and P. Legendre. Metric and euclidean properties of dissimilarity coefficients. *Journal of Classification*, 3(48), 1986.

[10] R. P. Hall. Computational approaches to analogical reasoning: A comparative analysis. *Artificial Intellegence*, 39:39–120, 1989.

[11] D. R. Jeffery and G. Low. Calibrating estimation tools for software development. *Software Engineering Journal*, 5(4):215–221, 1990. 0268-6961.

[12] R. Jeffery, M. Ruhe, and I. Wieczorek. Using public domain metrics to estimate software development effort. In *Seventh International Software Metrics Symposium*, pages 16–27, 2001.

[13] G. Kadoda, M. Cartwright, L. Chen, and M. Shepperd. Experiences using case-based reasoning to predict software project effort. In *EASE: Evaluation and Assessment in Software Engineering Conference*, Keele, UK, 2000.

[14] C. F. Kemerer. An empirical validation of software cost estimation models. *Communications of the ACM*, 30:416–429, 1987.

[15] J. Keung. heoretical maximum prediction accuracy for analogy-based software cost estimation. In *Asia Pacific Software Engineering Conference*. IEEE, 2008.

[16] J. Keung and B. Kitchenham. Optimising project feature weights for analogy-based software cost estimation using the mantel correlation. In *Proceedings of 14th Asia-Pacific Software Engineering Conference*. IEEE, Dec 2007.

[17] J. Keung, B. Kitchenham, and R. Jeffery. Analogy-x: Providing statistical inference to analogy-based software cost estimation. *IEEE Transactions on Software Engineering*, 34(4), 2008.

[18] C. Kirsopp and M. Shepperd. Making inferences with small numbers of training sets. *Software, IEE Proceedings-[see also Software Engineering, IEE Proceedings]*, 149(5):123–130, 2002. 1462-5970.

[19] B. Kitchenham. *Software Metrics: Measurement for Software Process Improvement*. NCC Blackwell, Oxford, 1996.

[20] R. J. A. Little and D. B. Rubin. *Statistical Analysis with Missing Data*. John Wiley & Sons, NY, 2002.

[21] G. E. R. Lloyd. *Polarity and Analogy: two types of argumentation in early Greek thought*. Cambridge University Press, Cambridge, 1966.

[22] C. Mair and M. Shepperd. The consistency of empirical comparisons of regression and analogy-based software project cost prediction. In *2005 International Symposium on Software Engineering*, page 10 pp., 2005.

[23] E. Mendes and B. Kitchenham. Further comparison of cross-company and within-company effort estimation models for web applications. In *10th IEEE International Symposium on Software Metrics (METRICS'04)*, pages 348–357, 2004.

[24] E. Mendes, N. Mosley, and S. Counsell. A replicated assessment of the use of adaptation rules to improve web cost estimation. In *International Symposium on Empirical Software Engineering (ISESE'03)*, pages 100–109, 2003.

[25] S. Morasca, L. C. Briand, V. R. Basili, E. J. Weyuker, M. V. Zelkowitz, B. Kitchenham, S. Lawrence Pfleeger, and N. Fenton. Comments on: "towards a framework for software measurement validation". *Software Engineering, IEEE Transactions on*, 23(3):187–189, 1997. 0098-5589.

[26] T. Mukhopadhyay, S. Vincinanza, and M. J. Pietula. Estimating the feasibility of a case-based reasoning model for software effort estimation. *MIS Quarterly*, 16:155, 1992.

[27] I. Myrtvelt and E. Stensrud. A controlled experiment to assess the benefits of estimating with analogy and regression models. *Software Engineering, IEEE Transactions on*, 25(4):510–525, 1999. 0098-5589.

[28] C. Schofield. Software support for cost estimation by analogy. In *6th European Software Cost Modeling Conference*. Rolduc, NL., 1995.

[29] M. Shepperd and G. Kadoda. Comparing software prediction techniques using simulation. *Software Engineering, IEEE Transactions on*, 27(11):1014–1022, 2001.

[30] M. Shepperd and G. Kadoda. Using simulation to evaluate prediction techniques [for software]. In *Seventh International Software Metrics Symposium*, pages 349–359, 2001.

[31] M. Shepperd and C. Schofield. Estimating software project effort using analogies. *Software Engineering, IEEE Transactions on*, 23(11):736–743, 1997. 0098-5589.

[32] M. Shepperd, C. Schofield, and B. Kitchenham. Effort estimation using analogy. In *International Conference on Software Engineering*, pages 170–178, Berlin, 1996.

[33] M. J. Shepperd. Case-based reasoning and software engineering. Technical Report TR02-08, Bournemouth University, UK, 2002.

[34] P. N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison Wesley, 2005.

[35] S. Vicinanza, T. Mukhopadhyay, and M. J. Prietolla. Software effort estimation: An exploratory study of expert performance. *Journal of Information Systems Research*, 2:243–262, 1991.

[36] S. Vicinanza and M. J. Prietolla. Case based reasoning in software effort case based reasoning in software effort estimation. In *11th International Conference on Information Systems*, 1990.

[37] F. Walkerden and R. Jeffery. An empirical study of analogy-basd software effort estimation. *Empirical Software Engineering*, 4:135–158, 1999.

[38] P. P. Wiener. *The Dictionary of the History of Ideas*. Charles Scribner's Sons, NY, USA, 1974.