

Predicting Software Development Project Outcomes^{*}

Rosina Weber, Michael Waller, June Verner, William Evanco

College of Information Science & Technology, Drexel University
3141 Chestnut Street Philadelphia, PA 19104
{rosina.weber,mwaller}@drexel.edu; {june.verner,william.evanco}@cis.drexel.edu

Abstract. Case-based reasoning is a flexible methodology to manage software development related tasks. However, when the reasoner's task is prediction, there are a number of different CBR techniques that could be chosen to address the characteristics of a dataset. We examine several of these techniques to assess their accuracy in predicting software development project outcomes (i.e., whether the project is a success or failure) and identify critical success factors within our data. We collected the data from software developers who answered a questionnaire targeting a software development project they had recently worked on. The questionnaire addresses both technical and managerial features of software development projects. The results of these evaluations are compared with results from logistic regression analysis, which serves as a comparative baseline. The research in this paper can guide design decisions in future CBR implementations to predict the outcome of projects described with managerial factors.

1 Introduction and Background

Software development project failure can be very costly. Risk factors that can determine project failure tend to become evident only in the later stages of software development life cycle; often too late to steer a project safely back on course. As a result, software project managers can be aided greatly by tools that can identify likely success or failure at an early stage, as well as those factors that may contribute to development problems. Our goal is to develop a tool that can make a good prediction of a project's outcome early in the development life cycle and indicate success or risk factors. This tool has to be flexible enough to accommodate managerial features and be able to manage a variety of knowledge tasks (e.g., capture, reuse) comprising a knowledge management (KM) system.

We chose case-based reasoning (CBR) for this tool because CBR is a methodology [1] that can support the flexible automation of the entire process. Additionally, CBR is appropriate for software development related tasks because the methodology resembles human judgment [2]. CBR also offers advantages to support KM efforts [4, 5, 6, 7]. CBR is characterized as a lazy learner that predicts the outcome of new

^{*} D. Bridge and K. Ashley (eds.) Case-Based Reasoning Research and Development. LNAI 2689, 595-609, Berlin Heidelberg:Springer-Verlag.

cases by using a k-nearest neighbor classifier, thus presenting some potential benefits to predict project success (e.g., a relatively small training cost and effort conjugated with an explanation for the classification [3]).

There are a number of different techniques that can be used to implement CBR; the current view is that certain design choices can bias the system's quality [8, 9]. Therefore, we need to test different techniques and select the one that performs best with the problem data. Our data was collected from 122 software developers who responded to a questionnaire about a software project they had recently worked on. The questions addressed both technical and managerial features of the chosen software development projects.

Past work using CBR for prediction of software development projects has focused on the technical, quantifiable aspects of software development, predicting, for example, development effort [2, 8, 9] rather than addressing qualitative managerial factors. In [9], Watson et al. compared three CBR techniques and found that the weighted Euclidean distance was the most accurate method for predicting development effort for web hypermedia software. In [2] Finnie et al. compared CBR techniques with linear regression analysis. They predicted effort represented by a continuous dependent variable, which is better suited for linear regression models. Kadoda et al. [8] also predicted software development effort with CBR and compared its performance with stepwise regression. They concluded that there is strong association between features of the dataset (e.g., training set size, nature of the cost function) and the success of a technique, and that the best technique can be determined on the basis of each dataset [8].

The use of CBR to help manage software engineering projects is commonly associated with the experience factory (EF) [10, 11] - a framework that structures the reuse of experience and products obtained during the software development life cycle. For example, Althoff et al. [7] have adopted the EF model to create an experience base to reuse experiences and products obtained in the development of CBR systems. Our approach differs because our core goal is to predict success and provide advice to software project managers. Our strategy is not intended to explicitly record methods employed in one development and enable detailed reuse, but simply to gather general experience in software development projects in order to identify success factors. In our approach, it is the user's responsibility to search for mitigants when a given factor seems to suggest potential failure. Our approach is more superficial and designed for easier implementation and acquisition.

When using CBR to manage or predict success of managerial projects, sometimes the number of features describing these projects may outnumber the number of cases. The way Cain et al. [11] dealt with this problem was by incorporating domain knowledge to choose the features to explain success or failure. They combined the notion of explanation-based learning (EBL), which uses domain knowledge to generalize a concept from a training example [13]. Their parameterized combination of CBR plus EBL is detailed in Subsection 3.3.

We tested different techniques to assess how well they performed with our data. After testing unweighted measures we used a hill-climbing feedback method to learn weights and prevent imperfect features from participating equally in predicting

project outcome. We then introduced and investigated a weighted version of the CBR plus EBL approach. As we identify the best similarity measure to provide accurate outcome predictions, we will introduce a method to use this technique to identify success factors with CBR. Because failure factors may have multiple mitigants, managers will have the information necessary to avoid project pitfalls.

We also used a non-CBR technique, logistic regression (LR) [14], to predict the outcomes for the same dataset. LR is generally considered the most accurate statistical technique for modeling dichotomous dependent variables, particularly in datasets with fewer than 100 examples, but it is also a method that tends to be expensive [14]. Cleary et al., explain that LR is a highly desirable statistical model and should be used for model fitting and hypothesis testing [14]. Hence, we use LR as a point of comparison for testing various CBR techniques.

This paper is organized as follows. Section 2 discusses the questionnaire used for data collection. Section 3 describes the prediction techniques we used. In Section 4 we evaluate the techniques used and describe our results. Section 5 extends the use of the most accurate CBR technique to identify software project success factors. In Concluding Remarks, we present our conclusions and future work.

2 Collecting Software Development Project Data

The collection method is crucial in obtaining reliable data. Once data is collected and understood by a KM system it can be managed to systematically benefit other projects. Verner [15, 16] is a software engineer who collected the data we used in this research to investigate software project critical success factors. The data was collected for the sole purpose of analyzing software development projects and identifying the factors that determine project success or failure.

Table 1 Examples of questions in the questionnaire

ID	Question
q2_1	What was the level of involvement of the customers/users?
q2_4	Were the customers/users involved in making schedule estimates?
q3_3	Was the scope of the project well defined?
q3_5	Did the customers/users make adequate time available for requirements gathering?
q3_7	Did the requirements result in well-defined software deliverables?
q4_2	Was the delivery date decision made with appropriate requirements information?
q4_8	Did the project have adequate staff to meet the schedule?
q4_11	Did the schedule take into account staff leave, training, etc.?

Removing all variables that describe features unknown early in a project resulted in a total of 23 variables. The questionnaire incorporates both objective and subjective human judgments about these projects. Table 1 provides some example questions,

which will be further discussed in this paper, referenced by their ID. The format of these answers are *yes/no* and multiple-choice.

This questionnaire addresses the areas of *management support*, *customer/user interaction*, *requirements*, *estimation* and *scheduling*, in relation to project outcome. The outcome section originally included questions that allowed for conflicting answers. The questionnaire asked for project success or failure from the perspective of the organization and from the perspective of the developer answering the question. We merged these variables into a single variable and removed all projects where outcome variables were in conflict. After eliminating the conflicting answers, the initial group of 122 project records was reduced to 88 – 67 describing successful projects and 21 describing failed projects.

3 Techniques to Predict Project Outcomes

Our goal is to design a system to systematically acquire software development project data and understand it within a KM framework. A case-based reasoner predicts the outcome of new projects in this system. In this section, we discuss the CBR system we implemented to evaluate different similarity measures to predict project's outcome, describe CBR techniques and LR.

3.1 CBR Implementation

Our CBR implementation uses the data gathered with the questionnaire discussed in Section 2, from which we created a case base with 88 cases described by 23 features and a binary variable for project outcome. This implementation entails the use of four subtasks of the retrieve CBR process, namely, *identify features*, *initial match*, *search*, and *select* [17]. These are standard implementations except for the *select* subtask. In order to accommodate situations in which multiple cases have the same similarity but different outcomes, we broke all ties by selecting the class of the case with the next highest similarity. We treat all features as symbolic; if they have the same value the similarity equals one, otherwise it equals zero; there are no intermediary degrees for similarity.

Currently, our implementation does not identify factors that contribute to success or failure. The results discussed later in this paper will lay the foundation for the development of a method to provide identification of these factors along with an outcome prediction. Suggesting countermeasures based upon these factors is an additional step to incorporate.

3.2 Unweighted k-NN

We first implemented CBR with a traditional unweighted k-NN classifier to serve as a baseline to compare with predictions developed with other techniques. This measure simply considers the number of similar features between a candidate case and a target case.

3.3 Explanation-Based Learning

Explanation-based learning (EBL) uses domain-specific knowledge to generalize a concept [13]. Cain et al. [11] used the EBL notion to explore feature relevance -in a combined approach with CBR - to classify a dichotomous dependent variable because the features in their dataset outnumbered the cases. We investigate their CBR+EBL approach because we are designing a tool with an evolving collection process, and thus having more features than cases is a future possibility. The work described in [11] has successfully classified foreign trade negotiation cases (50 cases, 76 features) by applying the CBR+EBL approach. The approach entails a parameterized similarity measure that incorporates both elements of traditional CBR (exploring the similarity of features between projects) and elements of EBL (exploring features' relevance supported by domain knowledge). The parameterized Equation (1) of CBR+EBL introduced in [11] is given by similarity measure S :

$$S = \frac{\sum_{i=1}^n \alpha * sim(case_i, cue_i) + \beta \sum_{i=1}^n relevance(case_i) * sim(case_i, cue_i)}{n + \left(\beta \sum_{i=1}^n relevance(case_i) \right)} \quad (1)$$

To implement the CBR+EBL approach, we need to acquire domain-specific knowledge and choose appropriate values for α and β .

Knowledge acquisition for EBL. This knowledge elicitation process is aimed at determining a relevance factor 1 or 0 for each possible question answered in the questionnaire. These factors represent, respectively, whether or not the answers influence the project's outcome. Thus, a factor 1 is given when the answer for a question is such that it supports the outcome of that specific project. For example, software engineering knowledge mandates that, to be successful, projects should have a schedule (consequently the lack of a schedule could explain failure). Therefore, there are two possible answers for the question asking whether the project had a schedule: yes or no. The final factor 1 or 0 can only be obtained when we also consider the given project's outcome. Thus, in a project with outcome of success, if the answer to the question about the existence of the schedule is also yes, then the relevance factor is 1 ; because, based on domain knowledge, we can state that this answer supports the outcome of success. The relevance factors vary as shown in Table 2.

We interviewed two software engineers for consistency and represented the knowledge with simple rules. For example, Figure 1 shows a multiple-choice question from the *customer/user* section of the questionnaire, and its associated rule obtained through knowledge acquisition from experts.

Table 2. Relevance factors for question about existence of schedule

Does the project have a schedule?	Project outcome is	Relevance Factor is
Yes	Success	1
Yes	Failure	0
No	Success	0
No	Failure	1

By implementing rules for all features where an association occurs based on domain-specific knowledge, we were able to determine the relevance factor of a particular feature for a case. Some features can be left without rules because no association was evident. For example, the Yes/No question q1_7, “Did senior management impact the project in any other way?” does not directly assess the type of impact and thus was the only feature in our dataset left without a rule. This question exemplifies how we use EBL to help evolve the collection method; by confirming the importance of a given question through its impact on project outcomes (question q1_7 will be removed or reworded in the next version of the questionnaire).

<p>Question:</p> <p>“2.1 What was the level of involvement of the customers/users? 1. none 2. little 3.some 4. reasonable level 5. high involvement”</p>
<p>Rule:</p> <p>IF ((outcome is success) AND (answer is either (4) OR (5))) OR IF ((outcome is failure) AND (answer is either (1) OR (2))) THEN relevance factor equals 1, otherwise relevance factor equals 0.</p>

Fig. 1. Example question and its associated rule

Experts were able to provide associations for nearly all questions because the entire questionnaire was conceived and designed with the sole purpose of analyzing software development projects. Considerable research has been published in this area [e.g., 16, 18].

An aspect that we did not implement concerns combinations of features. It is possible, and very likely, that experts using domain knowledge could find associations between two or more features. One association could explain a given outcome, while another may neutralize the effect of individual features. We did not extend our investigation to address this, although we will consider this aspect in future research.

Determining α and β . Equation (1) expresses similarity S as the weighted distance between two cases that incorporates β as a weight measurement for the relevance factor. In the EBL component, features are considered relevant when their values support the outcome of each case, and thus they are assigned a relevance factor. Therefore, different cases have different sets of relevant features. The CBR component of the equation, represented by α , explores the similarity of features between cases. The final step in applying the parameterized equation from [11] is to define values for α and β . The authors [11] who conceived the equation do not explicitly recommend values for α and β ; they used $\alpha = 1$ and $\beta = 15$. They stated that the equation is not very sensitive to β and that β values greater than one will produce similar results.

The baseline for these parameters occurs when $\alpha = 0$, in which case only the EBL component is evaluated and different values for β do not impact the accuracy. When $\beta = 0$, however, the equation evaluates simple feature counting (unweighted k-NN). We also evaluated the sensitivity of variations of these two parameters with respect to our data. Initially we set $\alpha = 1$, and then 5, 10 and 20 and varied β in search of the greatest accuracy. We found the maximum accuracy to occur in multiple pairs of points for α and β . To account for the sensitivity of these parameters in different datasets, to ensure consistency of the results, and given that the authors in [11] claim that values above one for β do not impact the results, we chose to use four different pairs for α and β : (1,1), (5,7), (10, 11.5), and (20, 32). Though we perform all the tests using all the pairs, we will present only the results obtained with the first pair (1,1).

Given the bias imposed by applying the same effect to all features (some may be irrelevant) [19] on the case-based component of the formula, we next investigate variable feature weights. Our goal is to extend the CBR+EBL approach (Equation (1)) by adding a representation of relative feature relevance to that awarded by domain knowledge. Hence, once we have the feature weights for all the variables, we implement the combined version of the parameterized equation:

$$S' = \frac{\sum_{i=1}^n w_i * \text{sim}(\text{case}_i, \text{cue}_i) + \beta \sum_{i=1}^n \text{relevance}(\text{case}_i) * \text{sim}(\text{case}_i, \text{cue}_i)}{\sum_{i=1}^n w_i + \left(\beta \sum_{i=1}^n \text{relevance}(\text{case}_i) \right)} ; (2)$$

where we use individual weights instead of α . To ensure the consistency of the results, we compute S' (Equation (2)) for $\beta = 1$ and $\beta = 15$. The results are discussed in Section 4.

3.4 Feature Weight Learning

The framework for feature weighting methods described in [19] suggests the use of incremental hill-climbing methods when the dataset contains interacting features. We

selected gradient descent (GD), which is a hill-climber that uses feedback from the similarity measure when examining each case.

GD can be implemented and modified by adjusting its geometric parameters. We used starting step size 0.5, ending step 0.02, step size update 0.9, and the number of cases tested was 10. These parameters resulted most effective in our preliminary tests. Having obtained weights to account for the relative importance of each feature, we used these weights to predict project outcomes using the weighted k-NN and weighted CBR+EBL.

3.5 Logistic Regression

LR is generally considered the most accurate and theoretically appropriate statistical technique for modeling dichotomous dependent variables, particularly in datasets with fewer than 100 examples, though it is also a method that tends to be expensive [14]. Cleary et al., explain that LR is a highly desirable statistical model and should be used for model fitting and hypothesis testing [14]. It is appropriate for our data because outcome is a dichotomous variable. LR produces a formula that predicts the probability of an occurrence as a function of the independent variables. LR overcomes the problem with linear models producing values for the probability outside the range of (0,1) desired for a dichotomous dependent variable [19]. Unlike ordinary least squares (OLS) regression, LR does not assume linearity of relationship between the independent variables and the dependent, does not require normally distributed variables, and in general has less stringent requirements with respect to the data.

4 Evaluation

First we want our evaluation to determine which of the CBR techniques performs best across three accuracy metrics when predicting project outcomes with our dataset. Second we want to compare the performance of the CBR techniques to LR for each metric. Third, we want to determine whether the weighted version of CBR+EBL (S') is more accurate than its unweighted version.

4.1 Methodology

We represent the performance of these techniques by using three metrics. *Accuracy* represents the number of correct predictions in relation to the total predictions. *True positives* represent the number of correct predictions of projects with outcomes of success. *True negatives* give the number of correct predictions of projects with outcomes of failure. In this paper, values of accuracy, true positives and true negatives are expressed as percentages.

The origin of the data is explained in Section 2. We used the data generated by the questionnaire and also prepared for LR. It has 23 symbolic features describing 88 training examples; 67 successes and 21 failures.

For the evaluation we used stratified sampling by randomly choosing six pairs of training sets (test sets were the complements), with 44 cases each, maintaining the overall proportion of positive and negative examples across all sets. Training sets 1, 3 and 5 have 33 positive and 11 negative examples; training sets 2, 4 and 6 have 34 positive and 10 negative examples. We will present our results in terms of the average and standard deviation across these six test sets.¹

LR and the weighted forms of CBR require the use of training sets. Training sets were used to either generate equations for LR or learn feature weights. Once completed, the training parameters were then tested on the testing sets, i.e. on data not included in training.

4.2 Results

A summary of our results is presented in Table 3. These results were obtained by applying unweighted k-NN, (unweighted) CBR+EBL, weighted k-NN, weighted CBR+EBL with $\beta=1$, and logistic regression (LR). The results are given with the average and standard deviations across the six test sets.

Table 3. Average and standard deviation for accuracy, true positives, and true negatives

Technique	Accuracy		True Positives		True Negatives	
	Ave.	St.Dev	Ave.	St.Dev	Ave.	St.Dev
Unweighted k-NN	76.9	1.7	85.6	2.1	49.6	9.2
CBR+EBL	80.7	3.4	85.6	6.2	65.4	10.0
Weighted k-NN	78.4	4.9	84.6	5.0	58.8	11.4
Weighted CBR+EBL	74.6	7.8	77.7	12.1	65.3	10.5
LR	80.5	5.3	88.9	12.3	29.1	34.3

On the *first hypothesis* evaluated, we found CBR+EBL to be the most accurate among the CBR techniques for all three metrics. It performed just as well as the baseline (unweighted k-NN) in predicting successful projects, and was superior in the other two metrics, though one should note that the standard deviation resulted from the CBR+EBL performance is higher than the baseline. The results also confirm the conclusions presented in [11] that CBR+EBL outperforms unweighted k-NN. These results support the conclusion to adopt CBR+EBL, particularly because we can expect to have case bases in which the number of features is greater than the number of cases, which seems to be no obstacle for CBR+EBL.

¹ Though we believe LOOCV is preferable, applying LR would require that we developed 88 sets of equations, therefore we relied on stratified sampling.

With respect to the *second hypothesis* evaluated, CBR+EBL slightly outperformed LR in the average accuracy, but LR presented a higher standard deviation. In the average of true positives, LR was superior. LR predicted successful projects well, but was not able to predict failed cases as accurately (less than 30% for true negatives). This was likely due to the sparsity of data among the group of failed software projects (i.e., 21 out of 88). Inclusion of additional failed project cases would very likely improve these results.

Given that the data for successful projects is sufficiently dense, the metric true positives in Table 3 emphasizes the loss in accuracy caused by using a combination of feature weights and the EBL measure (weighted CBR+EBL). This is the only technique not able to predict at least 80% of successful projects. Even the unweighted k-NN performs well, easily finding similar cases among positive examples. LR is the most accurate in this metric.

The relative lack of negative examples (or failed projects) makes the accuracy of CBR+EBL stand out. This is probably the reason why CBR+EBL tends to be more accurate in general. Even with fewer negative examples, it provides better predictions. Our results suggest that there is an advantage in using CBR especially CBR+EBL with this type of data, which may often be sparse in real world problems. Therefore, we recommend CBR+EBL to predict the (binary) outcome of software development projects.

For the *third hypothesis* evaluated, we wanted to compare the performance of the weighted version of CBR+EBL (S') with respect to the unweighted S . S' provided the lowest accuracy with the highest standard deviation. It actually performed more poorly in predicting successful projects from dense data than it did predicting failed cases from sparse data.

These results suggest investigating further why the weighted version of CBR+EBL did not perform well in accuracy and true positives but performed much better (with respect to other techniques) in true negatives. The variation in performance of LR suggests an association with the number of negative examples; hence we want to investigate if a similar association could be made.

Possible causes for performance of S' . Table 3 shows that the weighted version of CBR+EBL outperforms all techniques that do not use domain knowledge for the metric true negatives. This may suggest that the combination of both the CBR and EBL components would be appropriate to learn from sparse data (and predict failed projects). When we compare the performance of the unweighted versions alone, CBR+EBL performs (a little) better than k-NN for accuracy, exactly the same as k-NN for true positives, and (much) better than k-NN for true negatives. This comparison suggests that when we add the EBL component, the performance improves with respect to true negatives. Additionally, if we analyze the weighted versions, the weighted CBR+EBL performs worse than the weighted k-NN. It performs just as poorly for true positives and better only for true negatives. These facts, combined with the high standard deviations, found for the weighted CBR+EBL measure, instigate further examination. Given this preliminary analysis, our

hypothesis is that when we combine feature weights with the EBL component, it overestimates the relative importance of some features. This is detrimental to predictive accuracy with dense data, but when applied to sparse data, the method seems to work fairly well. In future work, we will perform a second experiment to evaluate this hypothesis; in this paper, we simply examine further our results.

In order to fully gauge the effect of combining feature weights with EBL, we examine the weights and the EBL component in different test sets. The weights were determined by gradient descent (see Section 3.4), while the EBL component is provided by the assignment of relevance factors (see Section 3.3). Table 4 ranks the six variables that received the largest number of relevance factors overall. The final row shows the performance of the weighted CBR+EBL for true positives for these sets. The two columns in Table 4 dedicated to the two test sets show each variable's overall ranking within each test set based upon their feature weights (the higher the number, the smaller the weight). For example, variable $q3_7^2$ was fourth among the variables overall in terms of the number of relevance factors assigned, ranked fourth in test set 2 and 16th in test set 6.

Table 4. Relevance factors and weights in test sets 2 and 6

Rank	Variable ID	Test Set 2	Test Set 6
1	q3_5	1	4
2	q3_3	5	19
3	q4_8	15	3
4	q3_7	4	16
5	q4_2	3	17
6	q3_8	7	10
True Positives	-	58.8	85.3

The analysis of Table 4 shows that five of the six variables were heavily weighted in test set 2 (rankings 1, 3, 4, 5, 7) and lightly weighted in test set 6 (rankings 10, 16, 17, 19). When comparing these rankings to the performance for true positives, it is clear that where the majority of these variables were heavily weighted, weighted CBR+EBL performed the most poorly and where these variables were most lightly weighted, CBR+EBL performed the most accurately. This supports our interpretation that prediction accuracy decreases when the same features receive high weights and are assigned relevance factors in the majority of the cases (overestimating the relative importance of some features). We will evaluate this hypothesis in future work, because if the combination of the two techniques increases prediction accuracy when the dataset is sparsely populated, this measure can be used in these cases.

² See meaning of feature in Table 1.

5 Associating Outcomes to Project Management

This paper's final challenge is to make use of the most accurate technique to perform an additional task. We would like to determine which factors have the strongest associations with particular project outcomes. These could then be highlighted as critical risk factors in projects heading for failure, allowing a project manager to identify key strengths and weaknesses early enough to establish corrective measures when needed.

Given the suitability of LR, we again use it as a benchmark. The LR process includes the identification of the variables that most strongly predict the dependent variable; thus, predictor variables are a byproduct of LR. Based on LR, the variables that have the most influence on project outcome are q3_3, q3_5, q4_2, and q4_8.

We use the parameterized equation S (CBR+EBL) to suggest predictor variables by predicting the outcome of each of the four questionnaire sections separately. We compare these prediction results with the prediction generated using the entire dataset. Isolated problem areas and features that predict outcome nearly as well as the entire dataset are assumed to be those most responsible for project outcome.

Table 5. Average accuracy (in %) for the four sections across the six test sets

Problem Area	Ave.	Std dev.
Management Support	70.08	7.39
Customer/User	77.65	1.71
Requirements	75.38	6.17
Estimation/Schedule	76.51	5.12

Table 5 shows the average accuracy in each of the four sections for the six test sets. Given the similarity of average accuracy of the three problem areas *customer/user*, *requirements*, and *estimation-schedule*, our strategy is to further examine these three areas for potentially useful predictors. We will exclude the *management support* section because of its lower accuracy when compared to the other sections.

In order to further investigate the three problem areas, we assess the frequencies of the relevance factors in each test set. Our assumption is that features that have scored a higher number of relevance factors are those most responsible for the project outcome. We note that these tests were performed on test sets with 44 cases, so that a feature that has, for example, been assigned a relevance factor 29 times, has influenced 65.9% of the cases. Table 6 summarizes the averages across the six test sets for the four most relevant variables in each of the three sections. These variables are the ones with the best potential to be predictors. Among these variables, q3_3 and q3_5 in the *requirements* section, and q4_8 and q4_2 in *estimation-schedule* are also the variables identified by LR.

Table 6. Average assignments of relevance factors

Customer-user		Requirements		Estimation and Schedule	
Variable ID	Ave	Variable ID	Ave	Variable ID	Ave

q2_1	29.5	q3_3	33.3	q4_2	30.0
q2_3	26.7	q3_5	33.5	q4_1list	22.5
q2_4	22.2	q3_7	30.3	q4_8	31.7
q2_6	25.7	q3_8	30.0	q4_11	22.0

We now examine the variable with the highest relevance factor frequencies (q3_5) to determine whether it has been valued because it is relevant to successful or failed projects. For this last analysis, we do not want to use averages, so we select test sets five and six because they are the sets that present relevance frequencies that are the most similar to the averages for the *requirements* section.

In test set 5, feature q3_5 supported successful outcomes in 87.9% of the cases, that is, 29 out of 33. It supported failure outcomes in 45.5% of the cases, 5 out of 11. In test set 6, q3_5 supported success in only 26 of 34 cases, which represented 76.5% of the cases. The number of failed projects supported by q3_5 in this set was 7, representing 70% of the cases (7 out of 10).

Looking exclusively at these two test sets, q3_5 supported success in over 80% of the cases; and supported failure in almost 60% of the cases. It would be premature to state whether this feature can be considered as a critical success factor in successful or failed projects. Further study is necessary to identify a method to validate such conditions for a variable. Because this variable deals with the time dedicated for requirements gathering by the *customer/users*, it is easy to accept it conceptually as a critical factor for both success and failure in this instance.

These tests are not conclusive but show promising results for further research into automatically determining success and risk factors. Further research will delve more deeply into methods for identifying critical factors for success and failure; this will help project managers to better understand failure and how to increase chances of success by taking action early in software project development.

6 Concluding Remarks

This paper extends our understanding of the amenability of using CBR to support KM tasks for managing knowledge from projects described by managerial factors. We conclude that the parameterized CBR+EBL similarity measure [11] is the most accurate technique for this application. The second most accurate CBR technique, weighted k-NN, performed nearly as accurately as CBR+EBL overall, but it did not perform as well for predicting project failures because it appears to be less tolerant of sparse data sets. When the dataset was sufficiently dense, CBR+EBL performed at or above the level of all other methods and it stood out when the data was sparse. For this reason, the CBR+EBL measure is the best choice for this type of prediction.

With respect to the CBR techniques we used, one intriguing result was that the weighted k-NN performed similarly to CBR+EBL (*S*) in some instances but combining the two methods (*S'*) generally decreased prediction accuracy. We believe that this is because the two methods provide similar relative levels of importance to

the same data and a combination of these methods ultimately overstates the relative strength of these variables. In general, this may lead to less accurate prediction results. As discussed in Subsection 4.2, however, when we analyzed prediction for sparse data sets, it is possible that the combination of these methods will allow for a strong association between key factors and outcomes. It is also important to note, however, that even with sparse data, the combination of these methods did not outperform the CBR+EBL method alone. Probing this question more deeply will be a focus of future research.

The accuracy provided by logistic regression suggests its use as a benchmark for the prediction accuracy of a case base, when the dependent variable is dichotomous. These techniques cannot be considered as competitive alternatives for supporting a KM framework because LR poses higher engineering requirements, as it requires statistical expertise and a complex process of analysis. These tasks may be more easily performed using a CBR technique, which has less stringent engineering requirements. The primary cost benefit of CBR is manifested through automation – knowledge acquisition and system reuse may be fully automated so that staff members may use the tool without needing in-depth knowledge about the techniques used by the tool.

We have shown how to detect potential predictor variables for project success. Our work identifies q3_3, a well defined project scope; q3_5, customers/users making adequate time for requirements gathering; q4_2, a delivery decision (schedule) made with adequate requirements information; and q4_8, assignment of adequate staff to meet the project schedule, as the success factors in our dataset. It is noteworthy that CBR has found q3_5 to be the most important factor for both project success and failure. As project managers well know, estimation of a reasonable schedule is impossible without good requirements; good requirements and a well-defined project scope go hand-in-hand; and good requirements are essential for assigning adequate staff to a project schedule.

There are a few areas of future work that we wish to explore: evaluate the causes of the performance of S' , validate predictor variable identification, investigate the use of predictor variables to derive mitigants for project management, and analyze combinations of features to assign relevance factors in EBL. In the CBR implementation, we will incorporate the step that indicates the success or failure factors.

Acknowledgements. Rosina Weber's research is supported in part by the National Institute for Systems Test and Productivity at USF under the USA Space and Naval Warfare Systems Command grant no. N00039-02-C-3244, for 2130 032 L0, 2002.

References

1. Watson, I.: CBR is a methodology not a technology. *Knowledge Based Systems*, 12 (5-6) (1999) 303-308

2. Finnie, G.R., Wittig, G.E., Desharnais, J.M.: A Comparison of Software Effort Estimation Techniques: Using Function Points with Neural Networks, Case-Based Reasoning and Regression Models. *Journal of Systems Software* 39 (1997) 281-289
3. Aha, D. W.: Feature weighting for lazy learning algorithms. In H. Liu & H. Motoda (eds.): *Feature Extraction, Construction and Selection: A Data Mining Perspective*. Norwell, MA, Kluwer (1998) 13-32
4. Aha, D.W., Becerra-Fernandez, I., Maurer, F., & Muñoz-Avila, H.: (eds.) *Exploring Synergies of Knowledge Management and Case-Based Reasoning: Papers from the AAAI Workshop (Technical Report WS-99-10)*. Menlo Park, CA, AAAI Press (1999)
5. Watson, I.: *Knowledge Management and Case-Based Reasoning: a perfect match?* In *Proc. of the Fourteenth Annual Conference of the International Florida Artificial Intelligence Research Society* Menlo Park, CA, AAAI Press (2001) 118-122
6. Weber, R., Aha, D.W., Becerra-Fernandez, I.: Intelligent lessons learned systems. *International Journal of Expert Systems Research & Applications* 20 1 (2001) 17-34
7. Althoff, K.D., Nick, M., Tautz, C.: CBR-PEB: An Application Implementing Reuse Concepts of the Experience Factory for the Transfer of CBR System Know-How. In *Proc. of 7th German Workshop on Case-Based Reasoning*, Würzburg (1999) 39-48
8. Kadoda, G, Cartwright, M., Shepperd, M.: Issues on the Effective use of CBR Technology for Software Project Prediction. In Aha, D., Watson, I., (eds.): *Case-Based Reasoning Research and Development*, LNAI, 2080, Springer (2001) 276-290
9. Watson, I., Mendes, E., Mosley, N., Counsell, S.: Using CBR to Estimate Development Effort for Web Hypermedia Applications. In *Proc. of the Fifteenth Annual Conference of the International Florida Artificial Intelligence Research Society*. Menlo Park, CA, AAAI Press (2002) 132-136
10. Basili, V.R., Caldiera, G., Rombach, H.D.: Experience Factory. In J. J. Marciniak, (ed), *Encyclopedia of Software Engineering* 1 (1994) 469-476
11. Jedlitschka, A., Althoff, K.-D., Decker, B., Hartkopf, S., Nick, M.: Corporate Information Network (COIN): The Fraunhofer IESE Experience Factory, IESE-Report No. 034.01/E. Version 1.0, May (2001)
12. Cain, T., Pazzani, M. J., Silverstein, G. Using domain knowledge to influence similarity judgment. In *Proc. of the Case-Based Reasoning Workshop*. Washington, DC, Morgan Kaufmann (1991) 191-202
13. Mitchell, T., Keller, R., Kedar-Cabelli, S.: Explanation-based generalization: A Unifying View. *Machine learning* 2 (1986) 47-80
14. Cleary, P.D., Angel, R.: The Analysis of Relationships Involving Dichotomous Dependent Variables, *Journal of Health and Social Behavior* 25 (1984) 334-348
15. Verner, J. M, Overmyer, S. P. and McCain, K. W.: In the 25 Years Since the Mythical Man-Month What Have we Learned About Project Management? *Information and Software Technology* 41 (1999) 1021-1026
16. Procaccino, J.D., Verner, J.M., Overmyer, S. P., Darter, M.: Case Study: Factors for Early Prediction of Software Development Success, *Information and Software Technology* 44 (2001) 53-62
17. Aamodt, A., Plaza, E.: Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *Artificial Intelligence Communications* 7(1) (1994) 39-59.
18. Reel, J.S.: Critical Success Factors in Software Projects, *IEEE Software* 16(3) (1999) 18-23
19. Wettschereck, D., Aha, D.W.: Weighting features. Veloso, M., Aamodt, A. (eds). *Case-Based Reasoning Research and Development*, LNAI 1010, Springer-Verlag (1995) 347-358

20. Whitehead, J.: Willingness to Pay for Bass Fishing Trips in the Carolinas. In An Introduction to Logistic Regression, Writing up results. (1998) Available online (last visited 03/31/2003): <http://personal.ecu.edu/whiteheadj/data/logit/>