Identifying Critical Factors in Case-Based Prediction

Rosina Weber*, William Evanco*, Michael Waller*, June Verner**
*College of Information Science and Technology, Drexel University, Philadelphia, PA 19104, USA
**School of Computer Science and Engineering, The University of New South Wales
{rweber, william.evanco}@cis.drexel.edu; mawaller@sbsglobal.net, jverner@cse.unsw.edu.au

Abstract

A reversible outcome is one that can be changed. For example, the failure of an ongoing project may be avoided if certain actions are taken, while an outcome such as the path of a hurricane cannot be changed under current knowledge. The major benefit of predicting reversible outcomes resides in the possibility to avoid unwanted results. For this purpose, it is necessary to identify contributing factors responsible for the outcome, which once modified, can steer the result to a desired outcome. Consequently, the incorporation of a method into a case-based reasoning system to identify contributing factors affecting an outcome can improve its usefulness. This paper compares different approaches, particularly the use of domain knowledge, with respect to their ability to identify sets of factors that reverse software development projects predicted to fail into a prediction of success.

Introduction

When predicting an outcome to a problem situation, the user might also want to know the reasons for the prediction in order to reverse an unwanted outcome. To take into account this user's perspective and consider the entire application context, case-based reasoning (CBR) systems can be equipped with the ability to identify factors responsible for a predicted outcome, which will increase their usefulness (Aamodt and Nygaard 1995).

Case-based prediction is typically performed with the use of cases consisting of real experiences. These cases tend to be sparse and biased (Kadoda, Cartwright and Shepperd 2001), but the incorporation of domain knowledge has been shown to provide valuable compensation (Cain, Pazzani and Silverstein 1991; Watson et al. 2002; Weber et al. 2003). Real cases represent anecdotal evidence of a probable prediction while domain knowledge offers scientific support.

There is enough evidence in the literature supporting the use of domain knowledge in case-based prediction, hence we want to explore its potential benefits to identify critical factors responsible for such predictions. For this purpose, we compare a number of different approaches and evaluate their abilities to reverse predictions in software development projects.

CBR systems have been used to explain events (Leake 1991) and identify reversible measures for failures (Kolodner 1993). For example, CHEF (Hammond 1986) included failures as part of the case and acquired reversible measures when failures occurred. This strategy has more chance to succeed in mature and stable domains, whereas in reasonably new fields of study, e.g. software engineering, or in highly subjective and social contexts, i.e. management, it would be difficult to guarantee the accuracy of the acquired reversible measures.

On the other hand, Leake's work (1991) on explanations although seeking to explain past events, and therefore is not concerned with reversible measures, seeks to link an explanation to a useful strategy based upon the explainer's goals. Analogously, preferable critical factors are the ones that are more likely to reverse unwanted outcomes, which is the user's goal.

Domain Knowledge in Case-Based Prediction

Case-based prediction has been applied to different problems. Some examples of reversible outcomes are legal cases (Brüninghaus and Ashley 2003), potential victims of crime (Redmond and Line 2003), expenses (Weber et al. 1996), cost (Stottler 1994), sales (Mcintyre, Achabal and Miller 1993), and projects (Cain, Pazzani and Silverstein 1991). In software engineering, CBR has been used to predict effort (Kadoda, Cartwright and Shepperd 2001; Watson et al. 2002), and success or failure (Weber et al. 2003).

The applications of case-based prediction tend to use real cases, which are usually scarce and thus expensive, leading to sparse case bases that do not generate good predictions. The use of domain knowledge has been suggested to compensate for sparse data particularly where features can outnumber the available cases. For example, Cain, Pazzani and Silverstein (1991) have demonstrated that a combined method of explanation-based learning (EBL) and CBR can produce more accurate predictions than CBR alone. One benefit of using this approach is that it can provide accurate predictions even when features outnumber cases because domain knowledge compensates for the lack of data. This same EBL+CBR approach was applied in Weber et al.

(2003), confirming its superiority with respect to traditional CBR. This method can also be useful for general prediction in relatively immature fields, where domain expertise cannot yet explain many occurrences. Another problem with case-based prediction (Kadoda, Cartwright and Shepperd 2001) refers to the existence of biases in specific design decisions (e.g., choice of similarity measure). The use of empirical methods represents a way to substantiate design choices (Watson et al. 2002); however, besides greater accuracy, the incorporation of domain knowledge has the potential to diminish the biases because it steers the retrieval and overall CBR performance to conform to domain knowledge. When using CBR for prediction, design decisions can impose bias, but adding domain knowledge tends to alleviate those biases.

The focus of this paper is to identify critical factors responsible for an outcome and to determine the role of domain knowledge in this task. This represents an additional aspect to take into account when selecting the technique to implement for case-based prediction. In the following, we briefly present the EBL+CBR approach that combines domain knowledge and experience for prediction.

Knowledge and CBR: The EBL+CBR Approach

The EBL+CBR approach introduced in (Cain, Pazzani and Silverstein 1991) combines both elements of unweighted CBR (similarity of features between cases) and elements of EBL (relevance of features according to domain knowledge). For example, in a successful software development project, whether or not requirements were gathered using a specific methodology is considered to indicate relevance to the project's result. Relevance is assigned in accordance with domain knowledge.

Implementing EBL+CBR. The implementation of this method includes a knowledge acquisition step. Each feature has to be laid out with its meaning and all of its allowable values. Domain experts have to answer, for each feature, whether each of its allowable values supports one or another outcome. Although this approach has been implemented by Cain, Pazzani and Silverstein (1991) and Weber et al. (2003) for binary classification, there is no reason, in principle, why this approach could not be implemented for a ternary classification scheme. The knowledge acquired from domain experts is represented in the system through rules. These rules assign relevance factors to each feature of each case. The original method uses crisp values 0 or 1 for relevance factors. The rules assess associations between feature values and outcome according to domain knowledge. For example, in a case describing the history of a patient with an outcome of emphysema, the attribute smoking having a value 'yes' results in the assignment of a relevance factor of 1. This means that domain expertise recognizes the value of this attribute as a contributor to the outcome.

This approach produces an interesting result when computing similarity: many cases end up having the same similarity value. This requires a special method to perform the retrieval's *Select* step (Aamodt and Plaza 1994) to determine the final prediction. Because this is an interpretive reasoner, the result is that more than one case can be used as references to a new case. For instance, the similarity assessment may find 3 cases above the threshold with the same similarity and they may not necessarily lead to the same classification. In this case, the system uses the mode to make the prediction.

Methods to Identify Critical Factors

When using a case-based prediction system, if the predicted outcome is different than the user desires, then critical failure factors can indicate problem areas which may be mitigated to reverse the predicted outcome. On the other hand, if the outcome is favorable, then the values of critical success factors can identify the strengths of the input case.

In this section we describe six methods to identify critical factors in prediction. These methods are based on knowledge, statistics or are instance-based. They also vary with respect to the scope of the factors they identify, some methods identify factors for the entire dataset whereas others are able to individualize factors for each project. The utility of methods that identify critical factors for an entire dataset is that they provide trends based upon a community of cases. When this community consists of real world experiences, they represent evidence of the importance of these factors.

The first approach to identify factors for the entire dataset is based on the *gradient descent (GD) method*. It is an instance-based method that is recommended to determine the relative importance of features in a dataset – and particularly to CBR - because it uses feedback from similarity to assign weights to each feature (Aha 1998). Our approach selects as critical factors all of those variables whose resulting importance values are above the overall average.

The second method is statistical and illustrates the strong association between critical factors with the prediction task. *Logistic regression* (LR) (Cleary and Angel 1984) is commonly used to predict the outcome of dichotomous variables. In LR, the dataset is examined to select features with the strongest correlations to the outcome and then these features are used for prediction purposes. Therefore, in LR, finding predictor variables is a requirement for prediction. LR also identifies factors in the entire dataset rather than in one specific project. LR selects predictor variables using a maximum likelihood approach.

Some of the approaches we discuss are derived from the EBL+CBR (Cain, Pazzani and Silverstein 1991) prediction method described above. This method has been implemented to predict the outcome of software development projects (Weber et al. 2003) and it is in this context that we present the remaining methods.

The third method was already introduced in Weber et al. (2003) as an alternative to identify critical factors in software development projects. It can be interpreted as a prediction-oriented or *feature-oriented method*. It identifies factors by measuring the prediction accuracy of a case-based prediction system and then submits each feature alone to the same system. The general idea is that critical factors will be the ones whose accuracy is closest to the overall accuracy of the dataset. More specifically, we adopt the EBL+CBR approach for the predictions and evaluate the metrics for true positives and true negatives with leave-one-out cross validation. We identify as success factors the features that produce accuracy closest to the overall accuracy of true positives and as failure factors the ones with overall accuracy closest to true negatives.

The fourth method also uses the EBL+CBR prediction system. It utilizes cases used in the prediction of each new project -- a genuine *case-based method* that can individualize the identification of critical factors. Before detailing how it works, we have to point out that because of the way that EBL+CBR computes similarity; more than one case may be used to generate a prediction. Thus, the number of cases actually used is one of its variables. The general idea is that if a previous case had, for example, a successful outcome and it is sufficiently similar to a new case to predict the outcome of this new case, then the features whose values are equal in both cases are success factors. Analogously, when the predicted outcome is failure, equal feature values suggest failure factors. When two cases are used for the prediction, we determine whether the value for each feature in the target case is either equal to all values in the two similar cases or different to all values in similar cases (interpreting the value as a failure factor in successful outcomes when

Table 1. Examples of 3 projects from the dataset

Case # Feature	83	92	115	4	14	35
PM authority	No	No	Yes	Yes	Yes	Yes
Initial commitment	Yes	Yes	Yes	Yes	Yes	Yes
Sponsors involved	No	Yes	No	Yes	No	Yes
SH involved	NA	NA	No	Yes	No	Yes
SM involved	No	No	Yes	No	No	No
EU involved	High	Some	Little	High	RSN	RSN
EU trust PM	Low	AVE	AVE	High	High	AVE
EU involved schedule	No	No	Yes	Yes	No	No
Good REQ	No	No	No	Yes	Yes	Yes
Realistic EU	No	NA	No	NA	NA	Yes
Any method REQGD	No	Yes	No	NA	No	No
ACCC REQ	No	No	No	No	No	No
Method REQGD	NM	INTW	NM	NM	NM	NM
Scope well defined	No	No	No	Yes	Yes	Yes
EU time REQGD	No	No	No	Yes	Yes	Yes
REQ central REP	No	No	No	Yes	Yes	No
REQ clear DLV	No	No	No	Yes	Yes	Yes
Size hurt REQGD	Yes	Yes	No	No	No	No
DD set with REQINF	No	No	No	Yes	Yes	Yes
How was DD set?	No DD	EXTE	CLLCMT	No DD	SM	PM
DP involved schedule	NA	NA	No	NA	NA	No
Enough staff	No	No	No	Yes	Yes	Yes
Schedule consult HR	No	No	No	No	No	Yes
Outcome	Failure	Failure	Failure	Success	Success	Success

Abbreviations: PM: project manager; SH: stakeholders; SM: senior management; EU: end users; REQ: requirements; REQGD: REQ gathering; ACCC: accurate and complete; DLV: deliverables; REP: repository; DD: delivery date; REQINF: information about REQ; DP: developers; HR: human resources and employee related issues; NA: Missing; NM: No Methodology; RSN: Reasonable; AVE: Average; No DD: No Delivery Date; INTW: Interview; EXTE: External Entity; CLLCMT: Collaboration committee different). We do not use values when using two cases and the new case's value is neither different from nor equal to both. The EBL+CBR approach can also produce sets of three cases actually used in the prediction. In those predictions, we also consider when the new case has feature values that are equal to the values of the majority of similar cases and different from the majority. The former values are interpreted as success factors while we consider the criticality to be inversely proportional in the latter.

Based on domain knowledge, we adopt the *knowledge-based* portion from the EBL+CBR approach to identify critical factors. First, we examine which features of the new case would have been assigned relevance factors by the EBL method (described in the previous section) with the predicted outcome. These features are the ones in which domain knowledge supports the conclusion that they are contributors to the predicted outcome. For simplification purposes, we assume that the reasoner predicts a binary variable. The dependent variable has allowable values, both positive and negative. If the prediction is a positive outcome, then the features that are assigned relevance factors for a positive outcome are success factors or failure factors otherwise. For the remaining features, we replace the predicted outcome to assign relevance factors for a negative outcome and then obtain the failure factors.

Finally, we combine the knowledge-based and the case-based methods by taking *the union* of the factors each individually identify. Our assumption is that cases can complement knowledge, particularly in immature fields. Next we compare these methods.

Preliminary Studies

In this study we want to compare the different methods described above and explore their strengths and weaknesses in identifying success and failure factors in the dataset described below. More specifically, we want to determine how the above methods perform in comparison to domain experts with respect to the ability of the success factors they identify to reverse predictions of failure in our dataset.

Dataset

The dataset used for this study consists of 88 real cases describing software development projects. This dataset has 23 symbolic features describing 67 projects that have succeeded 21 that failed. For the identification of individual success factors, we have selected twenty projects. These projects have all originally failed and when submitted to the EBL+CBR prediction, they were predicted to fail. Table 1 shows 6 projects, 3 that failed and 3 that succeeded, as an illustration of the dataset.

Methodology

The methodology consists of 3 stages: 1) Identification, 2) Reversal, and 3) Prediction. Identification is the step where each method identifies critical factors either for the entire dataset or for the twelve selected projects. When these methods use case-based prediction, they use the EBL+CBR method described earlier. Given that all twelve projects are initially predicted to fail, the methods will identify exclusively critical success factors. The methods that identify critical factors have been described previously. For the study, we add the identification of factors by domain experts, in order to provide a realistic base for comparison.

Reversal is the second stage that is executed for all of the methods in the same fashion, including the ones based on knowledge. The method for reversal takes each factor and reverses the value given in each of the 12 projects. The reversal of the value uses the same categorization learned in the elicitation of knowledge for the rules, which establishes values that are aligned or not with an outcome. For example, if the factor identified refers to the availability of enough staff in the project, a value corresponding to *yes* is converted into *no*, and vice-versa. This reversal step does not evaluate the quality of the converted value for the reversal. This means that the reversal is executed regardless of the intention of success. We decided on this approach in order to keep the methodology uniform for all of the methods.

Prediction is the last stage; it examines the quality of the identified factors by assessing their ability to reverse failure predictions into success. Note that all twelve projects have actually failed and were predicted to fail when submitted to the case-based prediction system. Therefore, we conclude that a set of factors, whose opposite values lead to predictions of success for the projects using the same prediction tool, are of good quality. Hence, each new project obtained after the conversion of its values is submitted to the EBL+CBR prediction system again to assess whether the factors are capable of reversing a prediction to fail into a success.

Table 2. Reversed predictions by method

Method	Reversed projects		Factors	
	%	Abs.	Num.	Eff.
Domain experts	66%	8	6	1.3
Gradient descent	92%	11	11	1
Logistic regression	42%	5	4	1.25
Feature-oriented	58%	7	5	1.4
Case-based	0	0	3.25	0
Knowledge-based	0	0	6.66	0
Union	8%	1	8.5	0.1

Results

Table 2 shows the results of applying the methodology as follows. The first column shows the percentage of reversed projects out of the twelve projects studied. The second column lists the absolute number of reversed projects. The third column presents the number of factors identified by each method that targeted the entire dataset (integer values) and the average amount (fractional values) of factors identified per project by the methods that target each project. The fourth column shows the project-factor ratio, i.e. the number of reversed projects by factor. This is a first attempt towards defining measures of efficiency.

We analyze separately the results of methods that target the entire dataset and the ones that individualize each project (the rows highlighted in gray). The superiority of the gradient descent method with respect to the number of reversed projects is evident but it does not seem promising given the low efficiency demonstrated by the project-factor ratio. Examining GD's factors, we notice that most factors identified by GD were also identified by the other methods. However, only two factors--a well defined scope and end users having time for requirements gathering--were identified by all of them. The negative aspect of these methods, particularly the GD method, is the large set of factors. It does not seem realistic to expect that users will be able to reverse eleven aspects in a project.

To analyze the methods that individualize each project (last three in Table 2), we revised our methodology to allow these methods to use their knowledge to guide their reversal strategy. These methods indeed have the ability to use embedded knowledge to reverse only the values that are not supportive of success.

Table 3. Effect of knowledge-based reversal

Method	Reversed projects		Factors	
	%	Abs.	Ave.	Eff.
Case-based	0	0	1.9	0
Knowledge-based	16%	2	0.92	2.2
Union	8%	1	2.6	0.4

The results of this study are laid out in Table 3. Even employing knowledge-based reversal, these methods still perform poorly. The number of factors decreases, although their efficiency seems to increase, given the project-factor ratio. The knowledge-based method was able to reverse two projects, while in the first study it did not reverse any; showing that without the knowledge-based reversal, good values were replaced by bad ones.

Finally, we examine the last potential contribution of knowledge-based methods to the identification of critical factors. We investigate whether the use of knowledge-based reversal can be used to complement statistical and instance-based methods. For example, GD failed to reverse one project. However, when we perform knowledge-based reversal we find that it still cannot reverse that one project. More interestingly, some projects are no longer reversed. This also occurs for the other methods. For example, the union method has identified sixteen factors for project number 89. Using knowledge-based reversal, only one value is changed, namely the value for whether the size of the project had hurt the elicitation of requirements. The result is that after only one change, the case now is predicted to succeed. Interestingly, this same project was reversed when different values were changed. In fact, those were values that contradicted domain knowledge and thus they were not changed with knowledge-based reversal.

Discussion

The ultimate goal of equipping CBR systems with a method to identify critical factors is to maximize the usefulness of case-based prediction to its users. Keeping this in mind, it is unrealistic to consider methods that identify larges sets of factors, particularly in the context of software development projects, where the change of a single variable can represent a significant shift in resource allocation.

The number of factors has been decisive in choosing our strategy to examine potential success factors. We could have, instead, started by examining failure factors, but these resulted in larger sets, which is not practical as a final solution. However, it is a source that this dataset can offer that may be revisited.

Although most software engineering studies implement linear modeling (e.g., Verner and Evanco 1999, Khalifa and Verner 2000), it is likely that different factors responsible for a given prediction are interdependent. As we observed in our preliminary studies, different methods were able to reverse a project's prediction using different sets of factors, and one method reversed a prediction contrary to domain knowledge. These observations suggest that an optimized CBR system that accounts for an entire context may have the additional benefit to help uncover knowledge previously unknown.

Conclusion and Future Work

The incorporation of knowledge into case-based prediction has been proposed to alleviate the biases imposed by design decisions and to compensate for sparse case bases. We conclude that, currently, the use of domain knowledge in immature domains should be used in combination with other methods, in the pursuit of a minimal set of factors to reverse unwanted predictions. It seems that there is potential in combining knowledge, particularly instance based methods, to explore the space of factors and uncover relations between factors not yet studied. Additionally, the use of previous cases has the advantage of conveying contextual knowledge, which can also be helpful in uncovering interdependencies among factors.

Besides the study of the interdependence of critical factors, it is also necessary to define the level of reversibility of factors, e.g., using measures of efficiency of factors throughout the dataset and by project. Factors that are easy to reverse should receive priority.

In future work, we want to compare these contributing factors to other factors studied in CBR systems. For example, factors used in HYPO (Ashley 1990) to support arguments, issues in the IBP algorithm (Brüninghaus and Ashley 2003), and goal-based explanation (Leake 1991) are likely candidates. In order to improve case-based prediction, the first step is to incorporate methods to identify critical factors that are able to individualize the identification of factors, and that minimize the number of factors while maximizing their chances to reverse unwanted outcomes. Additionally, we would also like to incorporate methods to identify reversible measures and to determine their chances to reverse outcomes.

Acknowledgements

The authors would like to thank the anonymous reviewers who greatly contributed to the improvement of this work. R. Weber is supported in part by the National Institute for Systems Test and Productivity at the University of South Florida under the USA Space and Naval Warfare Systems Command grant no. N00039-02-C-3244, for 2130 032 L0, 2002.

References

Aamodt, A., and Nygaard, M. 1995. Different Roles and Mutual Dependencies of Data, Information, and Knowledge - an AI Perspective on Their Integration. *Data and Knowledge Enigneering* 16:191-222.

Aamodt, A., and Plaza, E. 1994. Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *AI Communications* 7(1): 39-59.

Aha, D. W. 1998. Feature Weighting for Lazy Learning Algorithms. Liu, H. and Motoda, H. eds. *Feature Extraction, Construction and Selection: A Data Mining Perspective*, 13-32. Norwell, Mass.: Kluwer.

Ashley, K. D. 1990. *Modeling Legal Argument: Reasoning with Cases and Hypotheticals*. Cambridge, Mass.:The MIT Press.

Brüninghaus, S., and Ashley, K. D. 2003. Combining Model-Based and Case-Based Reasoning for Predicting the Outcomes of Legal Cases. Bridge, D., Ashley, K.D. eds. *Case-Based Reasoning Research and Development*, LNAI 2689, 65-79. Berlin:Springer.

Cain, T.; Pazzani, M. J.; and Silverstein, G. 1991. Using Domain Knowledge to Influence Similarity Judgment. In *Proceedings of the Case-Based Reasoning Workshop*, 191-202. Washington, DC: Morgan Kaufmann.

Cleary, P. D., and Angel, R. 1984. The Analysis of Relationships Involving Dichotomous Dependent Variables. *Journal of Health and Social Behavior* 25: 334-348.

Hammond, K. J. 1986. Learning to Anticipate and Avoid Planning Problems through the Explanation of Failures. In *Proceedings of the Fifth National Conference on Artificial intelligence*, 556-560. Menlo Park, CA: AAAI Press.

Kadoda, G.; Cartwright, M.; and Shepperd, M. 2001. Issues on the Effective use of CBR Technology for Software Project Prediction. Aha, D. W. and Watson, I. eds. *Case-Based Reasoning Research and Development*, LNAI 2080, 276-290. Berlin:Springer.

Khalifa, M., and Verner, J. M. 2000. Drivers for Software Development Method Usage. *IEEE Transactions on Engineering Management*, 47(3): 360-369.

Kolodner, J. 1993. *Case-Based Reasoning*. Los Altos, CA: Morgan Kaufmann.

Leake, D. B. 1991. Goal-Based Explanation Evaluation. *Cognitive Science* 15(4): 509-545.

Mcintyre, S. H.; Achabal, D. D.; and Miller, C. M. 1993. Applying Case-Based Reasoning to Forecasting Retail Sales. *Journal of Retailing* 69(4): 372-398.

Redmond, M. A., and Line, C. B. 2003. Empirical Analysis of Case-Based Reasoning and Other Prediction Methods in a Social Science Domain: Repeat Criminal Victimization. Bridge, D., Ashley, K.D. eds. *Case-Based Reasoning Research and Development*, LNAI 2689, 452-464. Berlin:Springer.

Stottler, R. H. 1994. CBR for Cost and Sales Prediction. *AI Expert*, 9(8):25-33.

Verner, J., and W. Evanco 1999. An Investigation Into Software Development Process Knowledge. *Managing Software Engineering Knowledge*, 29-47. Aurum, A.; Jeffery, R.; Wohlin, C.; and Handzic, M. eds. Berlin, Germany:Springer-Verlag.

Watson, I.; Mendes, E.; Mosley, N.; and Counsell, S. 2002. Using CBR to Estimate Development Effort for Web Hypermedia Applications. *Proceedings of the Fifteenth Annual Conference of the International Florida Artificial Intelligence Research Society*, 132-136. Menlo Park, CA:AAAI Press.

Weber, R.; Pacheco, R.; Martins, A.; and Barcia, R. 1996. Using Typicality Theory to Select the Best Match. Smith, I. and Faltings, B. eds. *Advances in Case-Based Reasoning,* LNAI 1168, 445-459. Berlin: Springer.

Weber, R.; Waller, M.; Verner, J.; and Evanco, B. 2003. Predicting Software Development Project Outcomes Bridge, D., Ashley, K.D. eds. *Case-Based Reasoning Research and Development*, LNAI 2689, 595-609. Berlin:Springer.