Looking at Comparisons of Regression and Analogy-based Software Project Cost Prediction

Carolyn Mair and Martin Shepperd
Brunel University, UK
{carolyn.mair, martin.shepperd}@brunel.ac.uk

Abstract

OBJECTIVE – This paper builds on our previous research in which we found inconsistency within and between results in empirical studies of software engineering cost estimation which had compared regression and analogy techniques. To this end, this paper aims to determine why and how these inconsistencies occur. In addition, we attempt to provide a solution which might reduce inconsistencies in future.
METHOD – We retrieved a sample of 11 journal papers and 9 conference papers which compared both regression and analogy-based prediction methods. Analysis of these papers uncovered inconsistencies within and between results. From this starting point we concluded that researchers should look beyond asking "What is the best prediction system?" Thus we now consider the reasons for these seemingly conflicting results.
RESULTS – We found little consistency between studies both methodologically and in terms of outcome.
CONCLUSIONS – We propose that researchers adopt more rigorous and uniform protocols when empirically investigating project cost prediction methods and interpreting the results.

1. Introduction

Project cost prediction is an important and problematic element to successful software engineering. Despite being a research topic for over forty years, a general solution remains elusive. There exists a large body of empirical software engineering studies that compare various prediction methods. One might then assume that with so much evidence, combining and comparing results and thus developing theories would be straightforward. Although it is common that different studies investigating similar phenomena are likely to produce conflicting results [1], the reasons behind these inconsistencies are frequently less clear.

As researchers in empirical software engineering, we are primarily interested in investigating methodological efficacy or the issues that impact on productivity, quality or cost with the aim of building relevant theories for our discipline. In order to be applicable in practice, theories relevant to software engineering need be grounded in real world evidence. Furthermore, to build a solid empirical foundation, results from independent experiments need to be combined [2].

The current work has arisen out of an attempt to conduct 'a study of studies' [3], also known as a meta-analysis. A meta-analysis may be defined as a systematic and documented procedure to search and screen relevant studies, code results, and provide a quantitative summary of the findings [2, 4]. An underlying problem is that software engineering researchers and research groups typically apply different research methods from other research groups. For example, heterogeneity has been found in sampling methods, data collection, analysis and reporting techniques. In addition, the studies are

influenced by different measurement and environmental factors, publication bias and the 'file drawer problem'[5]. Furthermore, results are often generalized to an inappropriate population [6]. Any or all of these factors confound the comparison and combination results and compound the problem of theory building.

This study focuses on the same empirical studies used in [1] which we believe to be the first systematic comparison of empirical evidence for two competing project effort prediction systems. However, whereas previously we were concerned with the mere existence of inconsistencies, in this study we attempt to determine why and how the inconsistencies occur. The aim is to understand how to increase and promote the use of methodological homogeneity so that results from future empirical studies can be effectively compared and combined.

The remainder of this paper is organised as follows: Section 2 details related work including a summary of the methodology, results and conclusions from our previous paper, and related work which proposes guidelines designed to increase methodological homogeneity and protocols to ease replication. Section 3 proposes reasons for the inconsistencies in our sample. Section 4 outlines how rigorous protocols and guidelines could be applied so that theories based on real world evidence could be built.

2. Related work

Effort prediction systems such as expert judgement, statistical or machine learning approaches typically have a primary cost factor (size or function points) which is adjusted in accordance with a number of cost drivers which characterise the project and influence effort [7]. Researchers and practitioners are interested in finding the 'best' prediction technique, however this frequently varies among studies. To overcome this variability and reduce risk, some researchers recommend using at least two prediction approaches [8]. Yet despite adopting these recommendations, no converging results have been obtained [9, 10]. From our initial systematic literature search we found that the most commonly applied prediction methods for software effort prediction systems are based on regression [11] or analogy [12]. There are several variants of both regression and analogy. Thus we included in our study any least squares regression method, excluded robust and logistic regression, and defined analogy as a variant of a *k*-nearest neighbour algorithm. We do not describe these methods here but the interested reader is referred to [1] for details.

Notwithstanding the particular prediction method or methods used, the success of each is dependent on a number of factors: data set characteristics [13], whether the data are external or internal to the organisation for which the model is built [14], experimental design [11, 15] inherent characteristics of the particular method [16] and interpretation of the results [12, 17]. Some researchers [18, 19] have found a particular method superior only to find different results in later work [20]. These examples illustrate the lack of synthesis and consistency when attempting to determine whether regression or analogy is the superior prediction method.

We compared results within and between empirical studies that used both regression and analogy prediction techniques for software project cost estimation from all software engineering journals plus conference proceedings [1]. We analysed the results quantitatively and qualitatively and found a small inconsistency between the evidence. Quantitatively we found that: 45% supported analogy, 35% regression and 20% were undecided. Qualitatively we found that: 35% supported analogy, 25% regression and 40% were undecided (see Table 1). In order to determine any single variable, (e.g. date or publication type) was a significant predictor, we conducted a discriminant analysis. However, no significant level of discrimination between the variables was found.

Table 1: Published papers using both regression and analogy-based software project effort prediction techniques (adapted from [1])

N.B. Publications are abbreviated as follows: JSS=Journal of Systems and Software; TSE=Transactions on Software Engineering; ICCBR=International Conference on Case-Based Reasoning; ICSM=International Conference on Software Maintenance; ICSE=International Conference on Software Engineering; ESE=Empirical Software Engineering; IST=Information and Software Technology; METRICS=International Software Metrics Symposium; ISESE=International Symposium on Empirical Software Engineering

Study			Quantitative analysis Empirical result in support of each prediction method		Qualitative analysis
Author	**Year**	**Publication**	**Analogy**	**Regression**	**Support**
Finnie, Wittig et al.	1997	JSS	1	0	Strong support Analogy
Shepperd and Schofield	1997	TSE	30	3	Strong support Analogy
Finnie, Wittig et al.	1997	ICCBR	1	0	Strong support Analogy
Niessink and van Vliet	1997	ICSM	24	0	Support Analogy
Hughes, Cunliffe et al.	1998	IEE Proc. Software	14	9	Neutral
Myrtveit and Stensrud	1999	TSE	0	12	Support Regression
Briand, El Emam et al.	1999	ICSE	6	12	Neutral
Angelis and Stamelos	2000	ESE	2	1	Support Analogy
Mair, Kadoda et al.	2000	JSS	2	1	Neutral
Jeffery, Ruhe et al.	2000	IST	0	21	Neutral
Briand, Langley et al.	2000	ICSE	0	6	Support Regression
Burgess and Lefley	2001	IST	3	9	Neutral
Shepperd and Kadoda	2001	TSE	20	12	Support Analogy
Jeffery, Ruhe et al.	2001	METRICS	12	13	Neutral
Mendes and Mosley	2002	ISESE	0	54	Strong Support Regression
Wieczorek and Ruhe	2002	METRICS	23	9	Neutral
MacDonell and Shepperd	2003	JSS	1	5	Support Regression
Mendes, Watson et al.	2003	ESE	9	3	Support Regression
Mendes, Mosley et al.	2003	METRICS	9	3	Strong support Analogy
Mendes and Kitchenham	2004	METRICS	12	7	Neutral

3. Reasons for inconsistencies

Understanding the reasons for these inconsistent results is central to making progress in software engineering. We [13] suggested previously that data set characteristics are an important determinant of technique success. We found that different data set characteristics will favour different techniques. Nevertheless, in some cases [14, 21, 22] results were inconsistent despite utilising the same data set (and thus context variables) and the same prediction techniques. We also found inconsistencies across accuracy measures. For example, CBR outperformed LSR in 2 out of the 3 comparisons performed in terms of MMRE [21], whereas LSR outperformed CBR in both comparisons in terms of MMRE and Pred(25) [22]. Furthermore, despite a lower MMRE for CBR, Pred(25) was greater in comparisons with both regression models in [14].

Missing data are common in software engineering metrics. How these data are handled influences prediction accuracy. Although several imputation methods exist, no standard procedure has been adopted by software engineering researchers. We saw for example that in [21] and [14], projects with missing data were excluded, whereas in [22] missing data were replaced by 'random samples from the other projects' (p.864). Prediction models are commonly trained on a sample of the data set and tested on the remainder. The remainder is termed the 'holdout' sample. Again, no standard protocol is used to determine how large the training and testing sets should be. In these three studies using the same data set and the same prediction methods, the holdout sample size varied from 15% of the projects [21], 22% [22] and in [14] the entire data set was used to generate the regression models. This evidence suggests that data set characteristics and context variables cannot be the only factors responsible for inconsistencies in comparative accuracy of prediction techniques. The lack of standardisation in software engineering research methodology leads to heterogeneous sampling, measurement, and reporting techniques. Each has an impact on the outcome.

We now consider other factors that might be responsible in part for the inconsistencies, for example, superior expertise or interest in a particular method [23]. Frequently such biases are explicit in studies and the reader takes these into account before reaching his or her own conclusions. However, when combining and comparing results from a number of studies, the meta-analyst simply takes the individual published results and uses these as data points for the meta-analysis. In this way, individual biases may remain undetected. We also found that protocols were rarely described, which means that replicating original studies would be problematic.

4. Proposals for reducing inconsistencies

In order to reduce inconsistent results within and between studies which are comparing prediction techniques for software project cost estimation, several factors must be taken into account. We suggest that data set characteristics require careful consideration and inspection prior to analysis. Details such as where the data come from, how they were collected and the start and end year of the project are salient to the analysis. Ignoring these factors, or treating them equally when they are not, is likely to produce unreliable results. Furthermore, we suggest that authors include the residuals, the raw data they have analysed, or at least provide details of where the data can be obtained. Thus, if necessary, the data can be analysed again to obtain a standard, and maybe more useful accuracy measure than was provided in the original paper. In a previous study, we found 12 different accuracy indicators [24]. We would like to propose the use of a standard measure, preferably the residuals so that all other measures can be constructed. This would facilitate the straightforward comparison and combination of results and ultimately, the application of meta-analytic techniques.

We further propose that authors detail their protocol throughout the study so that their work can be replicated. This would allow future results derived from the same phenomena to be reliably compared and would lead to the body of knowledge software engineering is demanding. When data are missing, which is often the case in software engineering project data sets, data editing and imputation methods need to be clearly defined. Furthermore at present, sampling procedures are not often described.

Detailed sampling procedures would allow inferences to be drawn more exactly and appropriately. Thus the population to which the results could be generalised would be more accurately defined.

These proposals are mainly objective and if adopted the results clearly would be more standardised. However, a major issue we have come across is that of the 'pet technique' which influences results and is likely more sensitive to standardisation. We would like to suggest that in the abstract of their paper, authors indicate their area of interest and how they intend to compare this with other methods. This would at least give the reader some idea of bias in the paper and would allow him or her to reach a more informed conclusion about the results of the work.

5. Conclusions

We used results from our previous work which found inconsistencies with and between comparisons of prediction methods for software engineering project cost estimation. In this paper we have suggested why these inconsistencies might arise, and have proposed ways to reduce them in the future in order to produce more readily useful data from which a body of knowledge, and ultimately theories can be built.

Acknowledgement

This work is supported by the Engineering and Physical Sciences Research Council of the UK under grant GR/S45119.

References

[1] C. Mair and M. Shepperd, "The consistency of empirical comparisons of regression and analogy-based software project cost prediction," *Proceedings of the International Symposium on Empirical Software Engineering*, *ISESE'05,* 2005.

[2] J. Miller, "Can results from software engineering experiments be safely combined?" *Proceedings of International Software Metrics Symposium, IEEE METRICS,* 1999.

[3] J. S. Armstrong, "Principles of forecasting: A handbook for researchers and practitioners," *International Series in Operations Research & Management Science,* 2001.

[4] J. Miller, "Applying meta-analytical procedures to software engineering experiments " *Journal of Systems and Software,* pp. 29-39, 2000.

[5] R. Rosenthal, "The "file-drawer" problem and tolerance for null results," *Psychological Bulletin,* pp. 638-641, 1979.

[6] L. Pickard, B. Kitchenham and S. Linkman, "An investigation of analysis techniques for software data sets," *Proceedings of International Software Metrics Symposium, METRICS,* 1999, pp. 130-142.

[7] C. Mair, M. Shepperd and M. Jorgensen, "An analysis of data sets used to train and validate cost prediction systems," *Proceedings of the International Conference on Software Engineering, ICSE*, 2005.

[8] S. G. MacDonell and M. J. Shepperd, "Combining techniques to optimize effort predictions in software project management," *Journal of Systems and Software,* vol. 66, pp. 91-98, 2003.

[9] E. Mendes, N. Mosley and S. Counsell, "Using an engineering approach to understanding and predicting web authoring and design," *Proceedings of Hawaii International Conference on System Sciences, HICCS-34*, 2001.

[10] E. Mendes, N. Mosley and I. Watson, "A comparison of case-based reasoning approaches to web hypermedia project cost estimation," *Proceedings of World Wide Web Conference, WWW2, 2002.*

[11] F. Walkerden and R. Jeffery, "Empirical study of analogy-based software effort estimation," *Empirical Software Engineering,* vol. 4, pp. 135-158, 1999.

[12] M. Shepperd and C. Schofield, "Estimating software project effort using analogies," *IEEE Transactions on Software Engineering,* vol. 23, pp. 736-743, 1997.

[13] M. Shepperd and G. Kadoda, "Comparing software prediction techniques using simulation," *IEEE Transactions on Software Engineering,* vol. 27, pp. 1014-1022, 2001.

[14] E. Mendes and B. Kitchenham, "Further comparison of cross-company and within-company effort estimation models for Web applications," *Proceedings International Software Metrics Symposium,* pp. 348-357, 2004.

[15] I. Myrtveit and E. Stensrud, "A controlled experiment to assess the benefits of estimation with analogy and regression models," *IEEE Transactions on Software Engineering,* vol. 25, pp. 510-525, 1999.

[16] G. R. Finnie, G. E. Wittig and J-M. Desharnais, "A comparison of software effort estimation techniques: Using function points with neural networks, case-based reasoning and regression models," *Journal of Systems and Software,* vol. 39, pp. 281-289, 1997.

[17] L. Angelis and I. Stamelos, "Simulation tool for efficient analogy based cost estimation," *Empirical Software Engineering,* vol. 5, pp. 35-68, 2000.

[18] E. Mendes and N. Mosley, "Further investigation into the use of CBR and stepwise regression to predict development effort for web hypermedia applications," *Proceedings International Symposium on Empirical Software Engineering, ISESE*, 2002,

[19] R. Jeffery, M. Ruhe and I. Wieczorek, "Using public domain metrics to estimate software development effort," *Proceedings of the International Software Metrics Symposium,* 2001.

[20] E. Mendes and S. Counsell, "Early web size measures and effort prediction for web costimation," in Proceedings of 9th International Software Metrics Symposium, METRICS, 2003.

[21] C. Mair, G. Kadoda, M. Lefley, K. Phalp, C. Schofield, M. Shepperd and S. Webster, "Investigation of machine learning based prediction systems," *Journal of Systems and Software,* vol. 53, pp. 23-29, 2000.

[22] C. J. Burgess and M. Lefley, "Can genetic programming improve software effort estimation? A comparative evaluation," *Information and Software Technology,* vol. 43, pp. 863-873, 2001.

[23] D. J. Spiegelhalter, C. C. Taylor and J. Campbell, *Machine Learning, Neural and Statistical Classification.* Upper Saddle River, NJ, USA: Ellis Horwood, 1994.

[24] B. Kitchenham, L. Pickard, S. G. MacDonell and M. J. Shepperd, "What accuracy statistics really measure," *IEE Proceedings Software,* vol. 148, pp. 81-85, 2001.