Realistic Assessment of Software Effort Estimation Models

Boyce Sigweni Martin Shepperd Tommaso Turchi
Department of Computer Science
Brunel University London
UB8 3PH, United Kingdom
{boyce.sigweni, martin.shepperd, tommaso.turchi}@brunel.ac.uk

ABSTRACT

Context: It is unclear that current approaches to evaluating or comparing competing software cost or effort models give a realistic picture of how they would perform in actual use. Specifically, we're concerned that the usual practice of using all data with some holdout strategy is at variance with the reality of a data set growing as projects complete.

Objective: This study investigates the impact of using unrealistic, though possibly convenient to the researchers, ways to compare models on commercial data sets. Our questions are does this lead to different conclusions in terms of the comparisons and if so,are the results biased e.g., more optimistic than those that might realistically be achieved in practice.

Method: We compare a traditional approach based on leave one out cross-validation with growing the data set chronologically using the Finnish and Desharnais data sets.

Results: Our realistic, time-based approach to validation is significantly more conservative than leave-one-out cross-validation (LOOCV) for both data sets.

Conclusion: If we want our research to lead to actionable findings it's incumbent upon the researchers to evaluate their models in realistic ways. This means a departure from LOOCV techniques, while further investigation is needed for other validation techniques, such as k-fold validation.

Categories and Subject Descriptors

D.2.9 [**Software Engineering**]: Management—*Cost estimation*

Keywords

Software engineering experimentation, Software effort estimation, Cross-validation approaches

1. INTRODUCTION

EASE '16, June 01-03, 2016, Limerick, Ireland

DOI: http://dx.doi.org/10.1145/2915970.2916005

Given the substantial commercial and societal benefits that might derive from accurate and timely software project cost prediction the topic has been the subject of much research over more than 40 years. Typically the main focus is on effort since this is where the greatest costs and uncertainty reside. For a review see [4] or for a more recent review that focuses on the application of machine learning techniques [21]. The upshot of this sustained research effort is a plethora of competing predictive models that range from traditional off-the-shelf models such as COCOMO, through statistically derived models based on regression modelling to sophisticated machine learners based on ensembles of regressors.

The challenge for the research community — and for practitioners — is to determine which models are 'best', acknowledging that this may require a context sensitive answer *and* some sense of how well such models might perform *in action*. The latter question is particularly important if we expect our research to have much real world impact. We need also to be aware that the real world is somewhat different from the research laboratory. In particular, the popular approach to evaluating competing predictive models is based on historical data sets. These are then used to predict the unknown by artificially dividing the data set into training cases and 'unseen' test cases. There are two common approaches to model validation [8, 14].

First is k-fold cross-validation where the data are randomly divided into k folds and each fold is successively held out for testing and then entered back into the training set. Typically for cross-validation $k \ll n$ where n is the number of cases in the data set. The allocation to folds is random. A side effect of this random allocation is variance in the results. For this reason it is common repeat the validation j times yielding a $j \times k$-fold cross-validation procedure.

Second, is leave one out cross-validation (LOOCV) sometimes referred to as jackknifing. This is a special case of k-fold where the fold size is $k = n$ i.e., one fold per case. Thus, each case is successively removed from the training set and used as the test case and then returned to training set as the next case is used as the test case. This approach to validation has the pragmatic advantage over generalised k-fold in that it is deterministic.

It is self-evident that neither approach is a particularly good approximation of the real world situation where the size of training set grows over time as software projects are completed and added as cases to the training set [10]. The

next project to be initiated becomes the new target or test case. One might refer to this as grow-one-at-a-time (GOAT) validation.

The remainder of the paper is organised as follows. The next section reviews our understanding of these validation techniques together with their usage in software effort modelling. We then present a comparison of these approaches compared with the more realistic GOAT validation. The final section considers the significance of our findings in terms of research practice and in terms of practitioners being able to accept our research findings as actionable.

2. RELATED WORK ON SOFTWARE EFFORT MODELLING VALIDATION

It is normal practice to validate predictive models using historical data where the outcomes are known so that it is possible to test the model's prediction against actual outcome. In order to simulate a model predicting cases for which the outcome is unknown various cross-validation procedures are used. Two widely used methods are k-fold cross-validation and LOOCV.

In a classic paper Kohavi [8] compares k-fold, with a bootstrap and LOOCV. A key observation is concerned with both bias and variance since essentially any technique acts as an estimator of the true, but unknown, population statistic. Subsequently, Isaksson et al. [3] have pointed out the negative impact of using small samples in genera,l and problems with LOOCV in particular, since the large degree of overlap between the training and test sets leads to some loss of independence which results in "a complicated variance contribution that is difficult to analyse in detail mathematically". Nevertheless, LOOCV can be a practical choice due to its deterministic behaviour.

The conclusion of researchers such as Isaksson et al. [3] is that for single data sets it is difficult to determine how reliable a performance estimate is. Furthermore, simulation based analysis suggests that either approach are often unreliable for small sample sizes as commonly encountered in real world applications. Finally, constructing a confidence interval based on the holdout test set remains the only rigorous yet practical useful alternative for assessing and reporting predictive performance.

It should be noted that both Kohavi and Isaksson et al. were concerned with classifiers whereas our particular problem domain is for real-valued prediction. In addition, software projects commence and terminate at distinct points in time hence we might better think of the data as a pseudo-time series.

In the field of software cost modelling a number of researchers have commented on the temporal aspect of the data and the restrictions this *ought* to impose on validation schemes. An early example is Lefley and Shepperd [9] who when evaluating the ability of genetic programming algorithms to form effort prediction models ordered the training data chronologically, however, the study did not examine the question as to whether this, more realistic, form of validation led to different results. Sentas et al. [15] adopted a similar approach to validate their models based on ordinal regression. Again they did not directly study the impact of using data ordered by time.

More recently Lokan and Mendes [11] posed the question of how realistic are the traditional k-fold and LOOCV approaches and does this matter? They identified that having sorted the data into chronological order there are the choices of merely using a training set that grows over time and the possibility of employing a moving window so that older, and perhaps obsolete, data are discarded. These questions were explored in the setting of comparing within company and cross company estimation based on the ISBSG data set. Their conclusions were that it made no difference, however, this is in the context of a much larger and more complex study so it would seem worth revisiting this question.

MacDonell and Shepperd also addressed this question both within and between projects [12]. They reported significant differences between LOOCV and GOAT validation schemas, and even when using a moving window method — even though the latter being less relevant for our purposes. However, the analysis was conducted on a very small data set of only 16 projects.

To summarise, k-fold and LOOCV remain the dominant approaches to empirical validation of predictive models (both classifiers and regressors) by researchers throughout the field of machine learning. Software engineering is no exception. Neither of these approaches view the data as emerging over time which for many applications is perfectly reasonable but for software project effort prediction is not.

3. AN EXPERIMENTAL COMPARISON OF VALIDATION METHODS

We outline the experimental method we used by detailing the (1) validation schemes (2) feature weighting approaches (estimation techniques), (3) performance metrics and (4) data sets we used. Throughout this study an Intel CORE^{TM} i5 vPro^{TM} PC was used to run R [13] and the ArchANGEL tool[1]. The results obtained by our experiments are analysed using the Blind Analysis protocol detailed in [20] in order to limit the analysis bias and improve their reliability. Note the experiments were run by BS, the raw results blinded by TT and analysed by MS.

3.1 Choice of cross-validation scheme

We compare two validation schemes, namely LOOCV and the more realistic time-based approach we term GOAT which is described at the end of this section. We choose LOOCV as an example of k-fold cross validation since it has the useful property of being deterministic although computationally quite demanding for larger data sets such as the Finnish data set.

3.2 Choice of feature weighting approach

Three feature weighting techniques for analogy-based estimation are used, the first two in their default archANGEL implementation:

1. *CBR* - It might be thought of as the baseline well-established technique and has been applied since the mid 1990s [17]. All features are equally weighted to predict the effort.

[1]archANGEL is the most recent version of the ANGEL software tool for project prediction. It may be downloaded from http://tiny.cc/5t4fdx

2. *FSS* - While CBR uses all features equally weighted, FSS' feature weights are either $0 \vee 1$. FSS excludes features that do not contribute (irrelevant features) to the predicted value. It has generally been found to be an improvement over CBR [19].

3. *FSW* - It employs continuous, non-negative weights to predict a new effort using a more efficient algorithm to search for individual feature weights. FSW is an extension by [18] to the default ArchANGEL implementation.

These methods present a range of possible strategies for ABE. A trivial or naïve approach is included in order to determine the extent to which the more sophisticated techniques offer any value. Note, however, that the choices are made primarily to provide some modern predictive approaches in order to contrast our two validation schemes. It is not the purpose of this study to evaluate which of these choices is 'best' so in a sense the choices are arbitrary. We merely need vehicles to determine the impact of different validation schemes.

3.3 Choice of performance metrics

In order to assess the accuracy of cost estimation techniques, various performance metrics have been considered. Typically statistics such as MMRE, MdMRE, PRED(n) and Standardised Accuracy have been used as the accuracy statistics for prediction systems. MMRE is one of the most widely used evaluation criteria for measuring the performance of competing software prediction models nevertheless it is highly problematic in part due to the asymmetric behaviour of any z-score [7]. Consequently we use absolute residuals as a simple unbiased metric [16].

3.4 Choice of data sets

For our analysis we use two software project effort data sets (see Table 1) namely Desharnais [1] and so-called 'Finnish dataset' used in [5]. These data sets were chosen because they are:

- *Widely used*: Desharnais is one of the most widely used data set in order to estimate the software development effort. A review by Sigweni [19] on ABE models employing feature weighting, found that 15 out of the 19 selected studies used the Desharnais data set. The Finnish data set is also widely used in effort estimation studies such as [6, 5, 2].

- *Representative*: In order for the data sets to be representative it is helpful if they are of different sizes. Therefore, one of the data set should be small i.e., have a small number of cases and features while the other should be large — therefore having a large number of cases and features. The Finnish data set would be representative of larger software engineering data sets (408 cases and 44 features), since software project effort data sets usually contain relatively few cases, typically less than 50 features and almost invariably under 500 cases [6]. Therefore, the Finnish data set used in this study is at the large end of this spectrum whilst the Desharnais data set, with only 9 features and 77 cases, would represent small to medium data sets. Although these data sets are quite old, they are still widely used in the Software Engineering community [19].

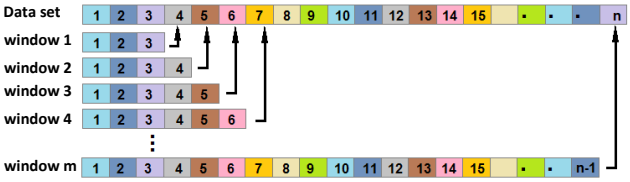

Figure 1: Validation data sets

The next two subsections provide detailed descriptions and explanation on how these data sets were processed.

3.4.1 *Desharnais data set*

The original Desharnais data set consists of 81 software projects obtained from Canadian software houses over 3 different development environments, published by Desharnais [1]. Unfortunately, 4 projects contained missing values therefore are excluded resulting in 77 projects used in this study. This data set is described by 9 attributes or features. These comprise one dependent attribute which is development effort measured in 'person-hours' (the feature to be predicted), and 8 independent attributes of which one the features (Language) is categorical while the rest are numeric. Outliers were not removed

3.4.2 *Finnish data set*

The Finnish data set is made up of project data collected by the software project management consultancy organisation STTF Ltd. The original Finnish data set had 90 features. The data set used in this study is the same data set used by [6], which removed some features due to missing values. The features are a mixture of categorical, continuous and discrete. The actual project effort is the dependent variable. As per the Desharnais data set we did not remove any outliers.

Out of 408 projects in the Finnish data set two cases had the actual effort attribute being equal to zero, therefore we removed these two questionable projects leaving 406 cases. This represents less than 1% of the total number of projects.

3.5 Methods for Accuracy Estimation

As mentioned in the introduction, the majority of software effort estimation studies use historical datasets to build and validate models for estimating software development effort. Almost all of them assign projects to training and testing sets without any consideration of the date in which the projects were completed [10]. Therefore, it is likely that the training set used to build a model to estimate the effort for a given target case includes cases that have not even started at the point the prediction of the target case is required [10]. However, in a real life setting, only completed projects can be considered when coming up with an estimate, i.e., one cannot consider future projects. Thus, there is an evident mismatch between normal industrial and research practice [10].

In contrast, GOAT validation consists of adding new projects to the data set as time passes; this approach better reflects real life settings, where completed projects can be used only once they're completed to predict a new

Table 1: Summary of the 'cleaned' data sets

Data set	No. of cases	No. of features	Min effort	Max effort	Mean effort	Median effort
Desharnais	77	9	546	23940	5046.31	3542
Finnish	406	44	55	63694	5031.00	2500

one. The data set looks like a growing window to the predictive mode (see Figure 1) and successively increases the size of the training set. This shows the projects in the case base ordered by their completion date. Therefore, case 1, denotes the earliest project (i.e., completed first). In terms of ABE, for example starting with 3 cases, window 1 would be used to predict the effort for case 4, and when case 4 is completed and its effort now established it is then added to the case base (window 2) and available to be used to predict the effort for case 5.

A number of researchers have considered using moving windows so the window is of fixed size m and as new cases are added, the most obsolete are dropped. A rationale for this approach is only the most current data is able to influence the predictive model [10, 12]. However, we view this as a particular type of modelling technique that chooses not to use all available data hence we don't explore this possibility further within the confines of this study. In other words, we view moving windows to be a particular prediction approach rather than a validation method.

First, both datasets were chronologically sorted by project completion date: whilst the Desharnais data set already provides this information directly as a feature (i.e., `YearFin`), the Finnish data set required us to add the duration of each project in the case base (i.e., `duration` expressed in months) to its start data (i.e., `DATESTART`) in order to compute a new feature we called `EndDate`, used only for the purpose of sorting and not to be included in the feature space used for prediction. Cases with the same end dates were then randomly sorted between each other to invalidate their original order.

We then constructed a new set of case bases using a growing window technique: (i) we selected the first 4 cases starting from the oldest one in both datasets (recall that this window has size 3, but we need an extra case to be the target); (ii) we then added the remaining cases one by one, drawing from the chronological order described before. Each sequence constructed by adding a new case is exploited as one single case base with the oldest cases as the knowledge base and the additional one as the target case, whose effort needs to be predicted.

This way we obtain 74 case bases for Desharnais and 403 for Finnish, each with a single target case to be predicted, requiring a total of 477 predictions to be validated.

4. RESULTS

Recall that this study uses blind analysis. Therefore, all statistical analysis of the results is based on blinded absolute residuals, that is they are provided with anonymised treatment labels to the analyst. However, to render the following discussion meaningful we remove the blinding.

A couple of factors complicate the analysis. First, the distributions of the residuals are extremely skewed and not amenable to simple transformations (see the boxplots of the absolute residuals for the validation techniques GOAT and LOOCV in Figure 2). Second, the Finnish data set has many ties. For this reason we use robust techniques.

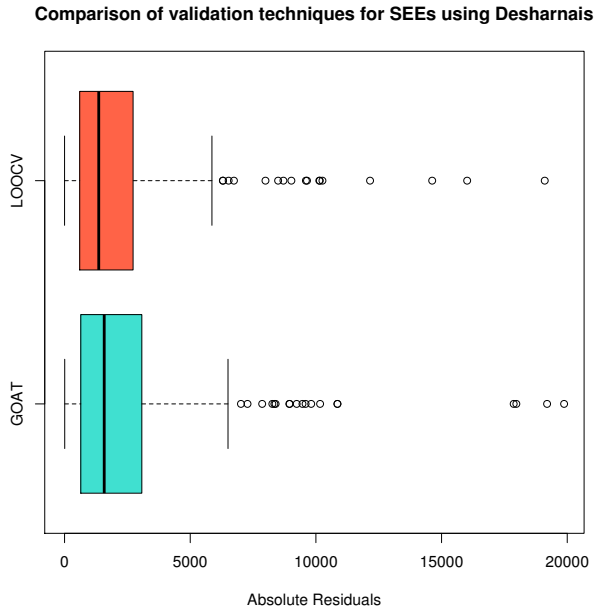

Figure 2: Boxplots showing residual distributions for two the validation techniques

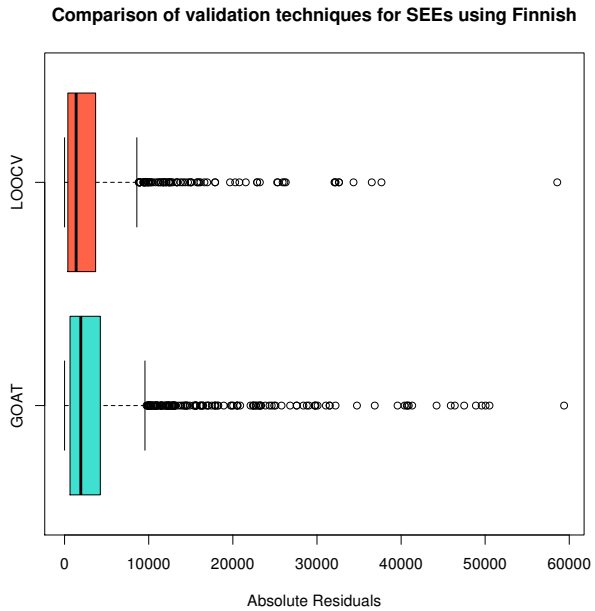

Figure 3: Boxplots showing residual distributions for two the validation techniques

Table 2: Comparing absolute residuals by validation technique using the Desharnais Data Set

Validation technique	Mean abs residual	Standard deviation	Median abs residual
GOAT	2455.8	2912.3	1576.4
LOOCV	2159.7	2547.8	1357.2

Table 3: Comparing absolute residuals by validation technique using the Finnish Data Set

Validation technique	Mean abs residual	Standard deviation	Median abs residual
GOAT	3919.9	6388.2	1923.3
LOOCV	2646.1	4174.8	1359.4

From Table 2 we observe that GOAT validation has a higher median absolute residual than LOOCV. This implies that for the same project and predictive model GOAT will give a more pessimistic view of likely performance. In addition, although both distributions are highly skewed (see Figure 2) we observe higher variance for the more realistic GOAT validation. This suggests that the performance of the predictive models may be less stable in a real world setting than implied by the artificial procedure of LOOCV.

Likewise, for the Finnish dataset (see Table 3) we observe a similar pattern. Again the LOOCV validation procedure yields seemingly better predictions and the evidence is somewhat stronger that the traditional cross-validation of LOOCV is optimistic. We also note higher variance (and standard deviation) for the realistic validation approach.

So the question is how significant is the factor of validation scheme compared with the second factor of feature weighting technique and do the two interact? To examine this we use a two-way robust ANOVA procedure based on 0.2 trimming of the means and bootstrapping. Since the design is within-within[2], we use Wilcox's `wwtrimbt` which is available in the R package WRS [22].

Table 4: Wilcox's Robust 2-way ANOVA with 0.2 trimming

Factor	Data set	p-value
Validation scheme	Desharnais	0.047
Feature Weighting technique	Desharnais	~ 0
Interaction	Desharnais	0.666
Validation scheme	Finnish	~ 0
Feature Weighting technique	Finnish	~ 0
Interaction	Finnish	0.046

Table 4 summarises the significance of each term in contributing to the variance of the response variable (absolute residuals) in a 2-way linear model with interactions. Although the choice of feature weighting technique dominates ($p \sim 0$) we see validation scheme also has significance ($p = 0.047$). This suggests that the results are impacted by how we validate.

In addition, for our set of three predictors, there is mixed evidence about whether the impact from the choice of validation scheme is influenced by the choice of feature weighting technique. For the Finnish dataset feature weighting technique has a highly significant association with validation scheme $p = 0.0460$. Interestingly, there doesn't appear to be any such interaction between the two factors for the Desharnais dataset. Therefore the results show that some prediction techniques (in terms of feature selection) may be influenced by the validation scheme but this doesn't always happen.

5. DISCUSSION AND CONCLUSIONS

In this study we have compared the behaviour of a traditional leave one out cross-validation (LOOCV) scheme with a time-based grow one at a time (GOAT) scheme. The motivation for this is the latter is far closer to what a practitioner would experience if they were to use a particular predictive model in reality. Recall, our questions are (i) does this lead to different conclusions in terms of the comparisons and (ii) does this lead to biased results e.g., more optimistic results than those that might realistically be achieved in practice?

Based on the three predictive models and two data sets in our experiment, the results strongly suggest that yes the choice of validation scheme *can* impact comparisons between models and that the differences between schemes are not symmetric. This means a LOOCV approach is biased with respect to the more realistic GOAT validation and distorts the results to make predictors appear more effective in the lab than they would perform if deployed in the field.

Our findings are in line with MacDonell and Shepperd [12] who reported that there were differences between LOOCV and a realistic validation scheme that accounted for data availability. However, this contrasts with Lokan and Mendes [11] who reported it did not make a difference. Even if the answer is it matters sometimes, this should be a concern for the research community since one can't tell *a priori* when it will be pertinent. In other words, if at least sometimes traditional validation schemes are optimistic but on other occasions there is no difference then the response must be to use a time based approach.

It is easy for researchers to neglect the needs of practitioners. In order for our research to be actionable it needs to be realistic and convincing. Even if one could make a case that in judging the likely performance of our predictive models it makes no difference whether the data are treated as a time series with a GOAT-like validation regime employed or a more traditional k-fold scheme one still might as well use the more realistic approach. Indeed we put it more strongly. If our goal is to conduct research that has impact upon professional practice then researchers need compelling reasons not to view data as a time series.

Given the significance of the research questions there is clearly a need for further work. It would be helpful to extend the study to also consider k-fold cross validation and to replicate it with other data sets and also different predictive models. An additional sophistication would be to ensure that a target project can only use data that are

[2]The design is within-within because each technique and validation scheme is applied to all the projects. This is also known as repeated measures.

available, i.e., projects that have completed at the target project's *commencement*.

Acknowledgements

The authors would like to thank Pekka Forselius for providing the Finnish data set. They would also like to thank the reviewers for their helpful remarks.

6. REFERENCES

[1] J. Desharnais. Analyse statistique de la productivitie des projets informatique a partie de la technique des point des fonction. Master's thesis, University of Montreal, 1989.

[2] Finnish Software Effort Dataset. http://dx.doi.org/10.6084/m9.figshare.1334271. 03 2015.

[3] A. Isaksson, M. Wallman, H. Göransson, and M. G. Gustafsson. Cross-validation and bootstrapping are unreliable in small sample classification. *Pattern Recognition Letters*, 29(14):1960–1965, 2008.

[4] M. Jørgensen and M. Shepperd. A systematic review of software development cost estimation studies. *IEEE Transactions on Software Engineering*, 33(1):33–53, 2007.

[5] C. Kirsopp and M. Shepperd. Case and feature subset selection in case-based software project effort prediction. In *The 22nd SGAI International Conference on Knowledge Based Systems and Applied Artificial Intelligence*, pages 61–74, 2003.

[6] C. Kirsopp, M. J. Shepperd, and J. Hart. Search heuristics, case-based reasoning and software project effort prediction. In *GECCO 2002: Genetic and Evolutionary Computation Conf.* AAAI, 2002.

[7] B. A. Kitchenham, L. M. Pickard, S. G. MacDonell, and M. J. Shepperd. What accuracy statistics really measure [software estimation]. 148:81–85, 2001.

[8] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *The AAAI International Joint Conference on Artificial Intelligence*, pages 1137–1145, 1995.

[9] M. Lefley and M. Shepperd. Using genetic programming to improve software effort estimation based on general data sets. In *GECCO 2003*, volume Lecture Notes in Computer Science, pages 2477–2487. Springer-Verlag, 2003.

[10] C. Lokan and E. Mendes. Applying moving windows to software effort estimation. In *3rd International Symposium on Empirical Software Engineering and Measurement (ESEM)*. IEEE Computer Society, 2009.

[11] C. Lokan and E. Mendes. Using chronological splitting to compare cross-and single-company effort models: further investigation. In *32nd Australasian Conference on Computer Science, Volume 91*, pages 47–54. Australian Computer Society, Inc., 2009.

[12] S. MacDonell and M. Shepperd. Data accumulation and software effort prediction. In *4th Intl. Symp. on Empirical Software Engineering and Measurement*. IEEE, 2010.

[13] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2013.

[14] J. Rodrõguez, A. Perez, and J. Lozano. Sensitivity analysis of k-fold cross validation in prediction error estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(3):569–575, 2010.

[15] P. Sentas, L. Angelis, I. Stamelos, and G. Bleris. Software productivity and effort prediction with ordinal regression. *Information and Software Technology*, 47(1):17–29, 2005.

[16] M. Shepperd and S. MacDonell. Evaluating prediction systems in software project estimation. *Information and Software Technology*, 54(8):820–827, Jan. 2012.

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

[18] B. Sigweni. Feature weighting for case-based reasoning software project effort estimation. In *The 18th International Conference on Evaluation and Assessment in Software Engineering*, page 54. ACM, 2014.

[19] B. Sigweni and M. Shepperd. Feature weighting techniques for CBR in software effort estimation studies: a review and empirical evaluation. In *The 10th International Conference on Predictive Models in Software Engineering*, pages 32–41. ACM, 2014.

[20] B. Sigweni and M. Shepperd. Using blind analysis for software engineering experiments. In *The 19th International Conference on Evaluation and Assessment in Software Engineering*. ACM, 2015.

[21] J. Wen, S. Li, Z. Lin, Y. Hu, and C. Huang. Systematic literature review of machine learning based software development effort estimation models. *Information and Software Technology*, 54(1):41–59, 2012.

[22] R. Wilcox. *Introduction to robust estimation and hypothesis testing (3rd Edn)*. Academic Press, 3rd edition, 2012.