Can Cross-company Data Improve Performance in Software Effort Estimation?

Leandro L. Minku, Xin Yao
CERCIA, School of Computer Science, The University of Birmingham
Edgbaston, Birmingham B15 2TT, UK
{l.l.minku, x.yao}@cs.bham.ac.uk

ABSTRACT

Background: There has been a long debate in the software engineering literature concerning how useful cross-company (CC) data are for software effort estimation (SEE) in comparison to within-company (WC) data. Studies indicate that models trained on CC data obtain either similar or worse performance than models trained solely on WC data.
Aims: We aim at investigating if CC data could help to increase performance and under what conditions.
Method: The work concentrates on the fact that SEE is a class of online learning tasks which operate in changing environments, even though most work so far has neglected that. We conduct an analysis based on the performance of different approaches considering CC and WC data. These are: (1) an approach not designed for changing environments, (2) approaches designed for changing environments and (3) a new online learning approach able to identify when CC data are helpful or detrimental.
Results: Interesting features of data sets commonly used in the SEE literature are revealed, showing that different subsets of CC data can be beneficial or detrimental depending on the moment in time. The newly proposed approach is able to benefit from that, successfully using CC data to improve performance over WC models.
Conclusions: This work not only shows that CC data can help to increase performance for SEE tasks, but also demonstrates that the online nature of software prediction tasks should be exploited, being an important issue to be considered in the future.

Categories and Subject Descriptors

D.2.9 [**Software Engineering**]: Management—*Cost estimation*; I.2.6 [**Artificial Intelligence**]: Learning—*Concept learning*

General Terms

Experimentation

PROMISE '12, September 21–22, 2012 Lund, Sweden

Keywords

Software effort estimation, cross-company estimation models, chronological split, online learning, concept drift, ensembles of learning machines

1. INTRODUCTION

There has been a long debate in the software engineering literature concerning how useful cross-company (CC) data are for software effort estimation (SEE) in comparison to within-company (WC) data. One of the reasons for that is that the number of projects available from a single-company is typically small, causing WC SEE models to perform poorly. If possible, it would be desirable to use CC data to improve the performance attained by models trained solely on WC data. Nevertheless, studies so far indicate that CC models obtain either similar or worse performance than WC models [7, 11, 13, 15, 14].

Most SEE research, including CC-related work [7], does not consider the chronology of the projects. Models are typically trained on a set of projects and evaluated on another set of projects, without considering whether the training projects were really available before the testing projects. Even though in reality estimations can only be performed considering previously existing projects, this type of evaluation strategy is still valid when dealing with stationary environments. However, SEE tasks are online (new completed projects keep arriving with time) and unlikely to be stationary, as software development companies and their employees evolve with time. For example, new employees can be hired or lost, training can be provided, employees can become more experienced, new types of software projects can be accepted, the management strategy can change, new programming languages can be introduced, etc. So, SEE models developed at a certain point in time may become obsolete and the chronology of the projects must be considered when developing/evaluating models.

Such environment changes affecting the underlying distribution of the problem are referred to as concept drifts [17] in the Machine Learning (ML) literature. A certain joint problem distribution can be called a concept. A recent WC SEE work by Lokan and Mendes [12] further supports the fact that SEE tasks suffer from concept drift. They reveal that using a certain WC subset composed of recent projects may provide better performance than using all previously existing WC projects. A few previous studies on CC versus WC models consider the chronology of data [11, 13, 15, 14], but they use all previously existing projects to estimate future projects, neglecting concept drift.

In this paper, we analyse the benefit of CC data when considering the online and changing nature of SEE tasks, as this could reveal previously unnoticed advantages of such data. The main research question answered by the paper is "Can Cross-company Data Improve Performance in Software Effort Estimation?" This question is further separated into:

- RQ1: Are CC data potentially beneficial for improving SEE? Under what conditions?
- RQ2: Can this potential benefit be used? How?

We answer RQ1 by showing that CC data can be beneficial or detrimental depending on the moment in time, as SEE tasks operate under concept drifting conditions (section 5). Such concept drifts are very particular to SEE environments, having each concept active for a relatively short number of projects. We find that CC data are potentially particularly helpful for this sort of concept drift. Our analysis also reveals some interesting features of data sets commonly used in the SEE literature.

RQ2 is answered by presenting and evaluating a new approach called Dynamic Cross-company Learning (DCL) to make use of CC data's potential for SEE (sections 6 and 7). The evaluation across multiple data sets shows that DCL successfully uses CC data to improve performance in comparison to the sole use of WC data. We also show that using an existing general purpose concept drift approach is not enough to significantly improve performance over multiple data sets in comparison to a WC approach not designed for dealing with concept drift.

The following contributions to SEE are provided:

- SEE concept drift features are examined and their particularities in comparison to usual concept drifts tackled by the ML literature are identified.
- CC data are revealed to be potentially beneficial for improving the performance of SEEs over WC models when considering the online and changing nature of these tasks. Concept drift can cause CC models to become more or less beneficial/detrimental.
- Interesting features of SEE data sets commonly used in the literature are revealed, showing that different partitions of CC projects present different usefulness and are better to be used separately than as a single set. Separating CC projects according to their productivity increases their potential benefit for estimating projects of a single-company.
- A new approach able to successfully use CC to improve performance over WC models is proposed.
- We show that the online changing nature of software prediction tasks should be further exploited, being an important issue to be considered in the next research frontier on software project estimation.

This paper is further organised as follows. Section 2 presents related work. Section 3 presents our formulation of SEE as an online changing environment problem. Section 4 describes the data sets used in the study. Section 5 presents an analysis to answer RQ1. Section 6 presents the proposed approach DCL. Section 7 presents an evaluation of DCL, which together with section 6 answers RQ2. Section 8 discusses threats to validity and implications to practice. Section 9 presents the conclusions and future work.

2. RELATED WORK

2.1 SEE Literature

Most work in the SEE literature does not consider the temporal relationship among projects and conclude that CC models obtain either similar or worse performance than WC models [7]. A few studies consider the chronology of the projects. Lokan and Mendes [11, 13, 15, 14] propose two types of SEE problem formulation considering chronology: project-by-project splitting and date-based splitting. The studies indicate that it is important to consider chronology, even though the conclusions regarding CC versus WC models did not differ much from previous work. CC models obtained either similar or worse performance in terms of Mean Absolute Error (MAE), which is a symmetrical measure not biased towards over or underestimates [14, 22]. The results in terms of the biased Z-measure further suggest that the CC models were more likely to underestimate effort.

Even though these studies consider chronology, they do not consider the existence of concept drift, which may cause projects from different periods to become more or less useful throughout time. Their models are also trained on mixed data from the single-company and other companies, disregarding the fact that CC projects may belong to different concepts from the WC projects. A posterior work [12] shows that using a window of the most recent projects can provide better performance than using all the previous projects available. This is to our knowledge the first work to consider the existence of concept drift in SEE. Nevertheless, it does not consider the issue of CC versus WC. It does not consider that older projects may become useful again in the future either, e.g., an experienced employee could leave the company, causing its productivity to become similar to the time before this employee was hired.

2.2 ML Literature

Approaches for dealing with online learning and concept drift in the ML literature typically assume the availability of large amounts of data, forming a data stream [21] unlikely to exist in SEE. Even though SEE concept drifts can be very particular (section 5.2.1), ML approaches for dealing with drifts might be helpful, even if not producing very good results when directly applied for SEE. It it is valid to investigate how these approaches perform and/or to use them as inspiration to create new SEE approaches.

An example of approach for concept drifting classification tasks is Dynamic Weight Majority (DWM) [10]. DWM creates different base models, each associated to a weight which is reduced when the base model gives a wrong prediction. Base models are dynamically added and removed, facilitating adaptation to drifts. DWM's predictions are based on the weighted majority vote among the base models. Additive Expert Ensemble (AddExp) [9] is an approach similar to DWM that works for both classification and regression tasks, even though regression is restricted to predictions in the interval $[0, 1]$. However, AddExp is likely to be less robust to noise. Another approach is Diversity for Dealing with Drifts (DDD) [19]. Its main idea is that very highly diverse ensembles (whose base models produce very different predictions from each other) are likely to present poor performance under stable conditions, but may become useful when there are drifts. So, DDD maintains a high diversity ensemble which is only activated upon drift detection.

3. FORMULATION OF THE PROBLEM

We formulate SEE as an online learning problem in which a new project implemented *by a single-company* is received as a training example at each time step, forming a WC data stream. Differently from typical online data stream problems [21], even though new projects arrive with time, the volume of incoming data is small. So, there are no tight space or time constraints. For instance, it is acceptable for a new model to be created from scratch whenever a new training project becomes available. Considering SEE's nature, it is likely that concept drifts occur. This assumption is shown to be correct in section 5.2.1.

We consider that there is a pre-existing set (or sets) of CC data. So, time steps are only counted for the WC data stream, not for the CC data. We leave the investigation of incoming CC data as future work. We assume that companies other than the single-company being estimated may represent different concepts. This is a reasonable assumption, as it is widely accepted that different companies are heterogeneous. The experiments shown in section 5.2.2 indicate that they indeed represent different concepts.

At each time step, after the model is updated with the new training project, the next ten projects of the WC data stream are estimated. We consider ten as reasonable value because not so many projects are produced per year by a company. Investigation of other values is left as future work.

4. DATA SETS

Five different data sets were used: ISBSG2000, ISBSG2001, ISBSG, CocNasaCoc81 and CocNasaCoc81Nasa93. These include both data sets derived from the International Software Benchmarking Standards Group (ISBSG) Repository[1] and the PRedictOr Models In Software Engineering Software (PROMISE) Repository[2]. Each data set uses projects from a particular company as the WC data stream, and projects from other companies to compose CC data sets. The term CC is used from this point onwards to identify only projects from other companies, excluding projects from the single-company.

4.1 ISBSG Data Sets

Three SEE data sets were derived from ISBSG Release 10, which contains software project information from several companies. The data were preprocessed, maintaining only projects with:

- Data and function points quality A (assessed as being sound with nothing being identified that might affect their integrity) or B (appears sound but there are some factors which could affect their integrity).
- Recorded effort that considers only development team.
- Normalised effort equal to total recorded effort, meaning that the reported effort is the actual effort across the whole life cycle.
- Functional sizing method IFPUG version 4+ or identified as with addendum to existing standards.

The preprocessing resulted in 187 projects from a single-company (WC) and 826 projects from other companies (CC). Three different data sets were then created:

[1]`http://www.isbsg.org`
[2]`http://promise.site.uottawa.ca/SERepository`

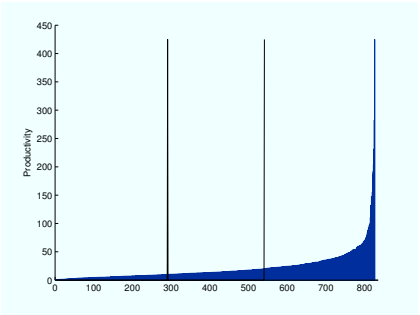


Figure 1: Sorted productivity for ISBSG CC projects. The vertical lines represent the ranges used to separate the projects into subsets.

Table 1: Ranges of productivity for CC subsets.

Data Set	Productivity Band	Number of Examples
ISBSG2000	[0,5]	56
	(5,13]	57
	(13,155.7]	55
ISBSG2001	[0,6]	72
	(6,14]	79
	(14,155.7]	73
ISBSG	[0,10]	291
	(10,20]	250
	(20,424.9]	285
Cocomo 81	[0,0.15]	24
	(0.15,0.35]	20
	(0.35,1.30]	19

- ISBSG2000 – 119 WC projects implemented after the year 2000 and 168 CC projects implemented up to the end of year 2000.
- ISBSG2001 – 69 WC projects implemented after the year 2001 and 224 CC projects implemented up to the end of year 2001.
- ISBSG – no date restriction to the 187 WC and 826 CC projects, meaning that CC projects with implementation date more recent than WC projects are allowed. This data set can be used to simulate the case in which it is known that other companies can be more evolved than the single-company analysed.

Four input attributes (development type, language type, development platform and functional size) and one output attribute (software effort in person-hours) were used. K-Nearest Neighbours [3] imputation was used for dealing with missing attributes for each data set separately.

The approach DCL proposed in this paper further uses a separation of CC projects into subsets. This was done according to their normalised level 1 productivity rate provided by the repository. The separation was based on the distribution of productivity. A representative example of productivity and its skewness is shown in figure 1. The ranges used for creating the subsets are shown in table 1 and were chosen to provide similar size partitions. This process could be easily automated in practice.

4.2 CocNasaCoc81

Cocomo Nasa and Cocomo 81 are two software effort estimation data sets available from the PROMISE Repository. Cocomo Nasa contains 60 Nasa projects from 1980s-1990s and Cocomo 81 consists of the 63 projects analysed by

Boehm to develop the software cost estimation model COCOMO [2] first published in 1981. Both data sets contain 16 input attributes (15 cost drivers [2] and number of lines of code) and one output attribute (software effort in person-months). Cocomo 81 contains an additional input attribute (development type) not present in Cocomo Nasa, which was thus removed.

Cocomo Nasa's projects were considered as the WC data and Cocomo 81's projects were considered as the CC data. The data sets provide no information on whether the projects are sorted in chronological order. The original order of the Cocomo Nasa projects was preserved in order to simulate the WC data stream. Even though this may not be the true chronological order, it is still useful to evaluate whether approaches are able to make use of CC data when/if they are beneficial. In order to create different CC data sets for DCL, the number of lines of code divided by the software effort in person-months ("productivity") was calculated for Cocomo 81. The productivity values are skewed, similarly to ISBSG's shown in figure 1. Projects were then separated according the ranges shown in table 1.

4.3 CocNasaCoc81Nasa93

This data set is also composed of Cocomo Nasa and Cocomo 81, but it uses an additional data set called Nasa 93, which contains 93 Nasa projects from 1970s-1980s and has the same input and output attributes as Cocomo Nasa and Cocomo 81. Even though both Cocomo Nasa and Nasa 93 are composed of Nasa's projects, they are used separately in the literature. For that reason, it is not known how useful Nasa 93 is for predicting Cocomo Nasa's projects and Nasa 93 was used here to compose a CC data set. Cocomo 81 was used as a single CC data set for DCL, instead of being divided into three. Similarly to CocNasaCoc81, the original order of the Cocomo Nasa projects was preserved in order to simulate the WC data stream.

5. POTENTIAL BENEFIT OF CC DATA

This section presents a novel analysis that reveals the potential benefit of CC data and answers RQ1. Sections 5.1 and 5.2 explain the experimental setup and analysis.

5.1 Experimental Setup

Regression trees (RTs) were used as the data models in the experiments, as they achieve good performance for SEE in comparison to several other approaches [18]. We used the REPTree implementation from Weka [6] in two ways:

- A CC-RT was trained offline on each CC data set. After that, at each time step, each RT was used to predict the next ten projects of the WC data stream.
- WC-RTs were created to reflect online learning. At each time step, the current RT was discarded and a new RT was trained on all projects so far (including the one received at the current time step). This RT was then used to predict the next ten projects of the WC data stream.

The CC-RTs were not used for learning projects of the single-company because, as explained in section 3, they may represent different concepts. Moreover, this allows us to use the performance obtained by each CC-RT for determining its potential benefit over the current WC-RT. If a certain CC-RT obtains better performance in a certain time step, it is potentially beneficial at this time step. If its performance is worse, it is detrimental.

The performance was calculated at each time step using the Mean Absolute Error (MAE) over the predictions on the next ten projects of the WC data stream. MAE is defined as $\sum_1^n \frac{|y_i - \hat{y}_i|}{n}$, where n is the number of cases considered, y_i is the actual value of the variable being predicted and $\hat{y}_i$ is its estimation. MAE was chosen for being a symmetric measure not biased towards under or overestimates, differently from other measures such as the Magnitude of the Relative Error (MRE) and the Z-measure [13]. Lower MAE indicates higher/better performance.

The parameters for the RTs were minimum total weight of 1 for the instances in a leaf, and minimum proportion of the variance on all the data that need to be present at a node in order for splitting to be performed 0.0001. These parameters were the most likely to produce good results in [18]. A single execution was performed for each data set, as we used deterministic RTs. The data sets used in the study were the ones presented in section 4.

5.2 Analysis

Figure 2 shows the MAE for each RT at each time step. Sections 5.2.1, 5.2.2 and 5.2.3 present interesting findings that reveal the potential benefit of CC data sets in SEE.

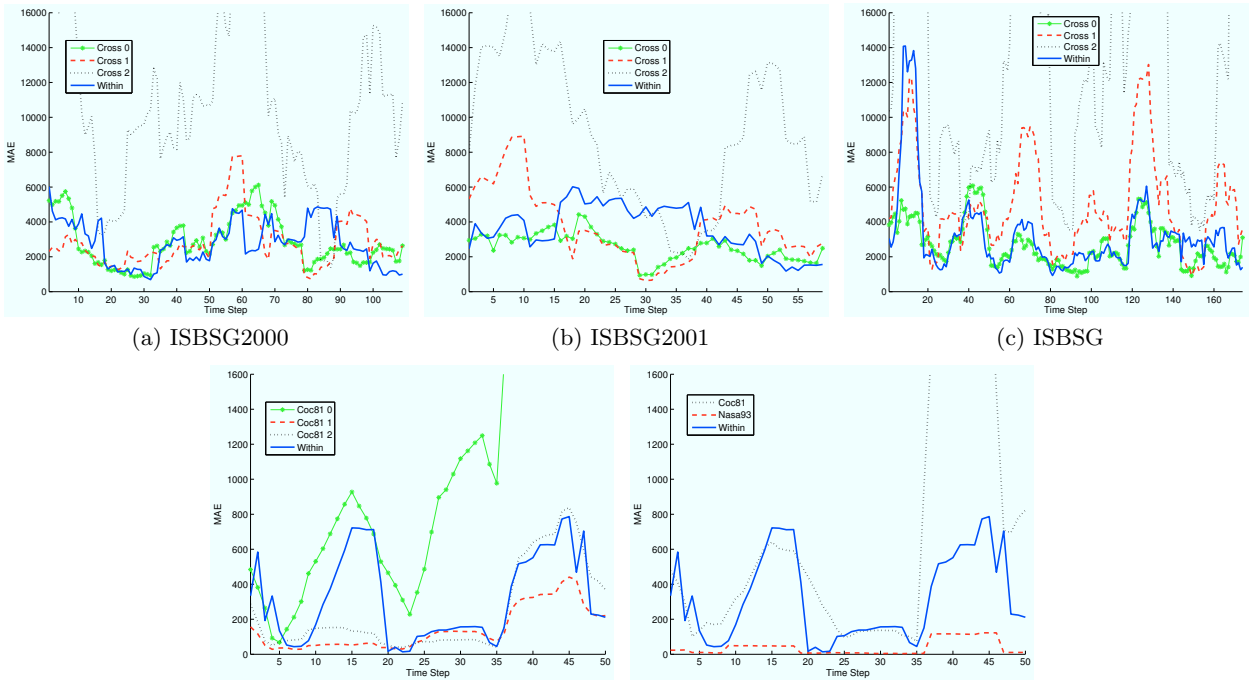
5.2.1 *Concept Drift in SEE*

The first point to be analysed regards the existence of concept drift in SEE. The data sets ISBSG2000, ISBSG2001 and ISBSG (figures 2(a), 2(b) and 2(c)) use the WC data stream in true chronological order, being the most relevant data sets for this part of the analysis. We can see from the graphs that the WC-RT does not present a general trend of always reducing its MAE. There are frequently periods in which the MAE increases before starting to decrease again. This is a strong indication of concept drift. The data sets CocNasaCoc81 and CocNasaCoc81Nasa93 use the original order of the Cocomo Nasa data set to simulate chronological order. The sudden changes in MAE (both increasing and decreasing MAE) suggest that Cocomo Nasa is likely to contain approximately one drift a every forth of the projects.

We can also observe that concept drifts in SEE have a key difference in comparison to the concept drifts usually tackled by the ML literature. Those approaches are designed to deal with data streams where input data arrive at very high rates, enough to cause storage and processing issues [1, 21]. In SEE, the incoming rate is typically very low. For instance, ISBSG2000 shows that in a period of around two years and a half, only 119 projects were made available for the single-company analysed. As a consequence, each different concept associated to the single-company is likely to be active for very few projects. For example, the concept likely to be active from time steps 1-15 in ISBSG2001 is associated to only 15 WC projects. Learning a new model using solely these projects would lead to very poor performance. So, simply applying existing concept drift approaches to learn the WC data may be somewhat helpful, but unlikely to provide very good results.

5.2.2 *Different Sets Representing Different Concepts*

The second point to be analysed regards the different concepts represented by the different CC subsets. Figure



(a) ISBSG2000 (b) ISBSG2001 (c) ISBSG

(d) CocNasaCoc81 (e) CocNasaCoc81Nasa93

Figure 2: Performance of offline CC and online WC-RTs for each data set in terms of MAE. Some RTs obtain very high MAE, but the limit of the y-axis was not increased to avoid hindering visualization of the better performing RTs.

2 shows that the performances of different CC-RTs for a particular WC data stream are considerably different from each other, suggesting that their corresponding CC subsets represent different concepts. Moreover, these concepts can become more or less beneficial/detrimental for the single-company whose projects are being estimated, depending on the moment in time. This is represented by the comparatively better/worse performance of the CC-RTs in comparison to the WC-RTs. This is a very interesting finding, considering that the literature so far has failed to use CC data to *improve* performance over WC models. It shows that it may be possible to use CC data to *improve* performance.

This behaviour is also intuitively reasonable. For example, a certain company may start adopting a new design process, becoming more similar to other companies that already used this process. Or the employees of a certain company may gradually become more experienced, so that this company would start behaving more similarly to more productive companies. It is worth noting that even though this behaviour matches intuition, capturing it through independent variables when creating a model can prove to be extremely difficult. The reason is that the number of independent variables which could potentially reveal concept drift is too large in comparison to the number of projects usually available for constructing a model. So, simply adding more variables to the model is not a feasible solution for this problem.

5.2.3 *PROMISE Data Sets Findings*

The last point to be observed in this section's analysis regards findings about the PROMISE data sets used in the study. Figure 2(d) suggests that even though part of Cocomo 81's projects (subsets 1 and 2) is beneficial for estimating Cocomo Nasa's projects, another part is detrimental (subset 0). In fact, the set of all Cocomo 81's projects together has very low potential for improving estimations of Cocomo Nasa's projects (figure 2(e)). This is an interesting finding because rather than using Cocomo 81 as a single data set, it may be worthier to separate projects according to their features and use only the most appropriate ones to estimate a certain project. This finding supports previous work that filters out projects likely to be detrimental for the SEE [8] and extends it to show that using the "correct" CC projects may provide better, rather than just similar performance to WC models in SEE.

Figure 2(e) further shows that models created using Nasa 93's projects provide better performance for estimating Cocomo Nasa's projects than using Cocomo Nasa's projects themselves. This is an intriguing fact that may be related to the quality of data collection.

6. MAKING BETTER USE OF CC DATA

Considering the analysis presented in section 5.2, an approach able to detect and successfully use CC data when they become beneficial is desirable and would provide an answer to RQ2. This approach should consider the fact that CC data may belong to different concepts, and that each concept of the WC data stream may be active for a short number of projects. In this section, we propose an approach called Dynamic Cross-company Learning (DCL), which considers these issues. A preliminary version of this approach called Dynamic Un+Reliable data learners (DUR) was presented in [16]. The differences between the two versions are explained, showing how DCL is better tailored to SEE.

DCL is inspired by the fact that models performing poorly at a certain moment may become beneficial in the event of

concept drift [17, 19]. So, it uses a fixed size memory of m models trained on pre-existing available CC data and one model specific for WC data stream learning. The CC models are never trained with projects from the single-company. Each model is associated to a weight representing how useful it currently is, similarly to [10, 9]. These weights are dynamically updated and allow DCL to emphasize estimations given by CC data models when they are useful.

DCL's estimations are based on the weighted average of the base models' estimations. However, the use of CC models is restricted/filtered. If the project to be estimated is "too different" from the projects used to build a CC model, this model is filtered out, i.e., is not allowed to contribute with the weighted average for estimating this project. This restriction was not used in the initial version DUR and was introduced to make DCL more tailored for SEE. Filtering out CC data for localization has shown to be useful for SEE [8].

The method to filter CC models out in DCL is as follows. The input variable project size is used to define whether a certain project is "too different" from the projects used to build a CC model. This variable was chosen for being likely to have the highest impact on the estimations and can be, for example, the functional size or the number of lines of code. In the absence of project size, another variable likely to be highly influential could be used. A CC model is then used for estimating a certain project only if the size of this project is lower than the quantile Q of the size of the training projects used to build this model, where Q is a pre-defined parameter. CC models that do not satisfy this requirement are filtered out of the estimation of this project, but are kept in the system so that they can possibly contribute to the estimation of other projects in the future. As the WC model is likely to be poor and unstable in the beginning of its life due to the small number of WC training projects, CC models only start being filtered out after a considerable number of WC projects have been presented (time step $> T_{start}$).

The separation of pre-existing CC data into m different sets is based on some a priori knowledge. For example, a set can be created for each different company, or different sets can be created based on different ranges of a certain independent or dependent variable. Separation based on ranges can be particularly useful and easily automated. An example of ranges separation is the productivity-based separation shown in section 5.2. The reason for the usefulness of productivity-based separation is that different productivity ranges simulate different possible concepts. When a concept drift happens, an organisation may become more or less productive, and CC models built using different productivity ranges could become more or less beneficial. Note that productivity is based on the effort. So, we cannot use the productivity of the project being estimated to decide what CC models are likely to be more beneficial for this project. Instead, DCL uses weights updated through its learning algorithm whenever a new WC project is completed in order to determine what base models are likely to be more or less beneficial at each time step.

Algorithm 1 presents the learning algorithm. Learning is divided into two stages: offline learning based on pre-existing CC training data and online learning based on WC training data stream. In the first stage, one base model is created to learn each CC data set in an offline way (line 3). As future work, the case in which CC data are also incoming

Algorithm 1 DCL

Parameters:
$D_c, 1 \leq c \leq m$: CC data sets.
β_c, β_w: factors for decreasing model weights $0 \leq \beta_c, \beta_w < 1$

1: {CC data learning:}
2: **for** each CC data set $D_c, 1 \leq c \leq m$ **do**
3: Create CC model L_c using D_c
4: $w_c = 1/m$ {Initialise weight.}
5: $w_{m+1} = 0$ {WC model weight.}
6: {WC data learning:}
7: **for** each new WC training project $p = (\vec{x}, y)$ **do**
8: $winner = \text{argmin}_{i,1 \leq i \leq m+1} |L_i(\vec{x}) - y|$
9: **for** $loser, 1 \leq loser \leq m+1 \wedge loser \neq winner$ **do**
10: $w_{loser} = \beta w_{loser}$, where
$\beta = \beta_c$ if $loser <= m$ and $\beta = \beta_w$ otherwise.
11: **if** p is the first WC training project **then**
12: $w_{m+1} = \frac{1}{m+1}$
13: Divide each weight by the sum of all weights
14: Use WC model L_{m+1} to learn p

will be investigated. Weights associated to CC models are initialized to $1/m$ (line 4). Weights associated to WC models are initialized to zero (line 5), so that DCL can be used for predictions before WC training projects become available.

The second stage of the learning is lifelong and one WC training project from the stream is received at each iteration, which corresponds to one time step. Each WC project is used for (1) weight update and (2) training the WC model. The WC model's training is done in the end of the iteration (line 14). It consists of using the incoming training project to train the WC model using its own learning algorithm, which may or may not need retraining on the previous projects.

The weight update rule is shown in lines 8–13. Each base model is used to perform an estimation for the incoming training project. The model with the lowest absolute error estimate is considered to be the winner (line 8). This can be either the WC or a CC model. The weight associated to all the loser models is multiplied by $\beta, 0 \leq \beta < 1$, where $\beta = \beta_c$ for the CC models and $\beta = \beta_w$ for the WC model. Lower/higher β values cause the system to quickly/slowly reduce its emphasis on models that are providing wrong estimations. The ideal values for β depend on the problem, but the median value of $\beta_c = \beta_w = 0.5$ can be used as a default value. The initial weight for the WC model is $1/(m+1)$ (line 12). After all weights are updated, they are divided by the sum of all weights (line 13).

This weight update rule is different from the one used in the initial version DUR, where a weight was multiplied by β if it performed an estimation considered as wrong. *Wrong* was defined as $MRE > Pred$, where $Pred$ is a pre-defined parameter. This rule has mainly two problems. Firstly, $Pred$ is a critical parameter dependent on the data set. An incorrect $Pred$ choice can cause all the weights to be reduced even if a certain model provided a better estimation than the others. The weights thus may not reflect the relative benefit of the models adequately. Secondly, MRE is asymmetric and biased towards underestimations, which are particularly problematic for SEE. Using MRE to update weights could emphasize models that give underestimations. Our new weight update rule overcomes these problems.

7. EXPERIMENTAL ANALYSIS

This section completes the answer to RQ2, presenting the experiments and analysis done to validate DCL. We also present a novel analysis on whether an existing approach prepared for dealing with concept drifts is able to improve performance for SEE in comparison to an approach not prepared for dealing with drifts. Sections 7.1 and 7.2 explain the experimental setup and analysis, respectively.

7.1 Experimental Setup

DCL and the following approaches were compared:

- RT trained on the WC data stream as in section 5.1. RT was used as a baseline approach to represent WC models not prepared for concept drift. As explained in section 7.1, RTs have shown to produce comparatively good performance for SEE [18].
- DWM trained on WC data stream (WC-DWM). WC-DWM was used to represent a concept drift handling approach from the ML literature [10]. It is included in the analysis to check whether it would be able to improve SEE performance in comparison to an approach not prepared for concept drift (RT).
- DWM first trained using the CC data as a stream and then trained on the WC data stream (CC-DWM). CC-DWM was used to verify whether using an existing concept drift approach to learn CC data would be enough to improve performance over WC models.

The base models used by DCL, WC-DWM and CC-DWM were RTs using Weka [6]'s REPTree implementation. The weight update rule used in WC-DWM and CC-DWM was the same as the one used in DCL, to provide a fair comparison and allow for regression tasks. A new base model is added to the DWM ensemble if its estimation on the current training project has absolute error higher than τ in a time step multiple of p. Existing base models with weight $< \theta$ are deleted also in time steps multiple of p.

The performance was measured in terms of MAE over the predictions on the next ten projects of the WC data stream. Additionally, the overall MAE across time steps was used to provide interpretable results in terms of the magnitude of the performance based on standardized accuracy (SA) and effect size Δ [22]. SA_L is viewed as the ratio of how much better L is than random guessing [22] and was based on 1000 runs of random guessing. Random guessing was defined as uniformly randomly sampling the effort over all the WC projects $p_i, 1 \leq i < t$, where p_t is the project being estimated. The effect size Δ used random guess as the control approach and was interpreted in terms of the categories [22]: small (≈ 0.2), medium (≈ 0.5) and large (≈ 0.8).

Seven different combinations of parameters were used for WC-DWM and CC-DWM: the default values of $\beta = 0.5$, $p = 1$ and $\theta = 0.01$, and all the combinations obtained by fixing two parameters to their default values and varying the remaining parameter using: $\beta \in \{0.3, 0.7\}$; $p \in \{10, 50\}$ for the larger data sets ISBSG2000 and ISBSG, and $p \in \{10, 15\}$ for the other data sets; and $\theta \in \{0.001, 0.1\}$.

The best performing parameters combination for each data set was used in the analysis, representing the best possible behaviour achievable by WC-DWM and CC-DWM considering these combinations. The parameter τ was set to $0.25y$, where y is the actual effort of the training project. We chose this value because effort estimations are frequently considered acceptable when they are within 25% of the actual effort [4], and choosing a value independent of the magnitude would be very difficult in practice. The need for setting τ is a disadvantage of applying this approach for regression.

The parameters used for DCL were the default values of $\beta_r = \beta_u = 0.5$, $T_{start} = 15$ and $Q = 0.9$. The parameters of the RTs for all approaches were the same as in section 5.

A single execution for each data set from section 4 was performed for DCL and WC-DWM, as they are deterministic when using the deterministic RTs from this study. The order of presentation of CC projects does not influence the results for DCL and WC-DWM. However, CC-DWM needs to use CC projects as a stream prior to WC learning. As the real chronological order for CocNasaCoc81 and CocNasaCoc81Nasa93 is unknown, thirty CC-DWM executions were performed with random CC order for these data sets.

The initial version DUR was preliminarily experimented using ISBSG, CocNasaCoc81 and CocNasaCoc81Nasa93 in [16]. That analysis was aimed at the general ML community, to show the advantages/disadvantages of DUR in comparison to an existing concept drifting approach (DWM) for learning either CC or WC data. The analysis presented in the current paper not only uses a new version of the approach better tailored for SEE, but also concentrates on important issues to the software engineering community, which were not investigated in [16]. These are the following. (1) We include RT as a control approach. As it is not known how well existing concept drift approaches perform for SEE, it is important to include an approach such as RT, which behaves relatively well for this task [16]. Including RT allows both evaluating WC-DWM and CC-DWM in the SEE context, and validating DCL for SEE tasks. (2) We analyse how well the models perform in comparison to random guess. This is important because SEE is typically a very difficult task and approaches might happen to perform worse than random guess [22]. (3) We provide interpretable results in terms of magnitude (SA) and effect size (Δ). This is important because companies can rely on this type of information to decide whether to deploy a new SEE model [22]. (4) The analysis presented here considers two additional data sets (ISBSG2000 and ISBSG2001). This inspired the use of a different statistical approach for the analysis, which is more adequate when comparing approaches using multiple data sets. The current paper also comments on the improvements achieved by DCL in comparison to DUR.

7.2 Analysis

7.2.1 Performance in Comparison to Random Guess

As it is irrelevant to compare approaches that perform worse than random guess against each other, we initially analyse the approaches against random guess. Table 2 presents the overall performance across time steps in terms of MAE and SA, as well as the Δ effect size against random guess.

We first determine the statistical significance of each approach's overall MAE across time steps in comparison to random guess. Wilcoxon rank sum tests using Holm-Bonferroni corrections considering all data sets and approaches at the overall level of significance of 0.05 were used for this purpose. The tests show that the overall MAE of all approaches is statistically significantly different from random guess.

Then, we verify the effect size Δ of this difference in per-

Table 2: Overall performance average across time steps. Cells in yellow (light grey) represent the best values. All approaches' MAEs are statistically significantly different from random guess according to Wilcoxon tests using Holm-Bonferroni corrections at the overall level of significance of 0.05.

MAE +- Std Dev				
Data Set	RT	WC-DWM	CC-DWM	DCL
ISBSG2000	2753.40 +- 1257.50	2862.50 +- 1256.80	2566.00 +- 1045.20	2352.60 +- 925.84
ISBSG2001	3622.00 +- 1368.00	3279.80 +- 1025.90	2934.10 +- 890.39	2873.10 +- 1235.90
ISBSG	3253.90 +- 2476.10	3160.90 +- 2508.50	3037.10 +- 2105.20	2805.60 +- 1468.20
CocNasaCoc81	319.46 +- 250.23	300.24 +- 278.23	244.94 +- 207.24	205.83 +- 214.70
CocNasaCoc81Nasa93	319.46 +- 250.23	300.24 +- 278.23	168.97 +- 188.14	109.82 +- 154.72
SA				
Data Set	RT	WC-DWM	CC-DWM	DCL
ISBSG2000	37.05	34.55	41.33	46.21
ISBSG2001	11.93	20.25	28.65	30.14
ISBSG	46.29	47.82	49.87	53.69
CocNasaCoc81	33.14	37.16	48.73	56.92
CocNasaCoc81Nasa93	33.14	37.16	64.63	77.01
Δ using random guess as the control approach				
Data Set	RT	WC-DWM	CC-DWM	DCL
ISBSG2000	2.57	2.40	2.87	3.21
ISBSG2001	0.96	1.63	2.31	2.43
ISBSG	1.34	1.38	1.44	1.55
CocNasaCoc81	0.57	0.63	0.83	0.97
CocNasaCoc81Nasa93	0.57	0.63	1.10	1.32

formance. RT and WC-DWM have effect size varying from medium to large, whereas DCL and CC-DWM always have very high effect size, showing a better behaviour.

Finally, we check the SAs. We can see from table 2 that the statistical tests and effect sizes are reflected on this performance measure. The SAs show considerably better performance than random guess in most cases, considering the SEE context and the difficultly of this task. RT and WC-DWM presented low SA for ISBSG2001, representing a clear need for improvement in this case.

7.2.2 *Overall Performance Across Time Steps*

As recommended by Demšar [5], Friedman statistical test was used for comparison of multiple models over multiple data sets. The measure compared was the overall MAE across time steps. The test detected statistically significant difference among the overall MAE of the approaches at the level of significance of 0.05 ($F_F = 58.5 > F(3, 12) = 3.490$, p-value = 0.0001). The ranking of approaches obtained from the test is shown in table 3. DCL was ranked first (lowest/best MAE) and CC-DWM was ranked second for all data sets. WC-DWM was ranked third and RT was ranked fourth for all data sets but ISBSG2000.

Average ranks by themselves provide a fair comparison of the approaches [5]. However, it is worth performing post-hoc tests to compare all approaches against our control approach RT. The z and p-values of the post-hoc tests [5] are shown in table 3. The tests over multiple data sets reveal that DCL's overall MAE is statistically significantly different from RT's, but the other approaches' are not. These results suggest that dealing with drifts decreases the overall MAE slightly (WC-DWM), and using CC decreases it a bit further (CC-DWM). However, the decrease only becomes statistically significant *over multiple data sets* when using DCL.

Table 4 shows the effect size of the difference in performance of each approach against RT. WC-DWM's Δ in relation to RT was always small, CC-DWM's was medium for two data sets, and DCL's was medium for two and high for one. These Δs reflect and confirm the results of the post-hoc tests, showing that it is worth to use DCL.

Table 3: Ranking average and standard deviation of approaches across data sets based on overall MAE; and z and p-values of the post-hoc tests for comparison of each approach against RT. The p-value in yellow (light grey) represents statistically significant difference of overall MAE using Holm-Bonferroni corrections at the overall level of significance of 0.05.

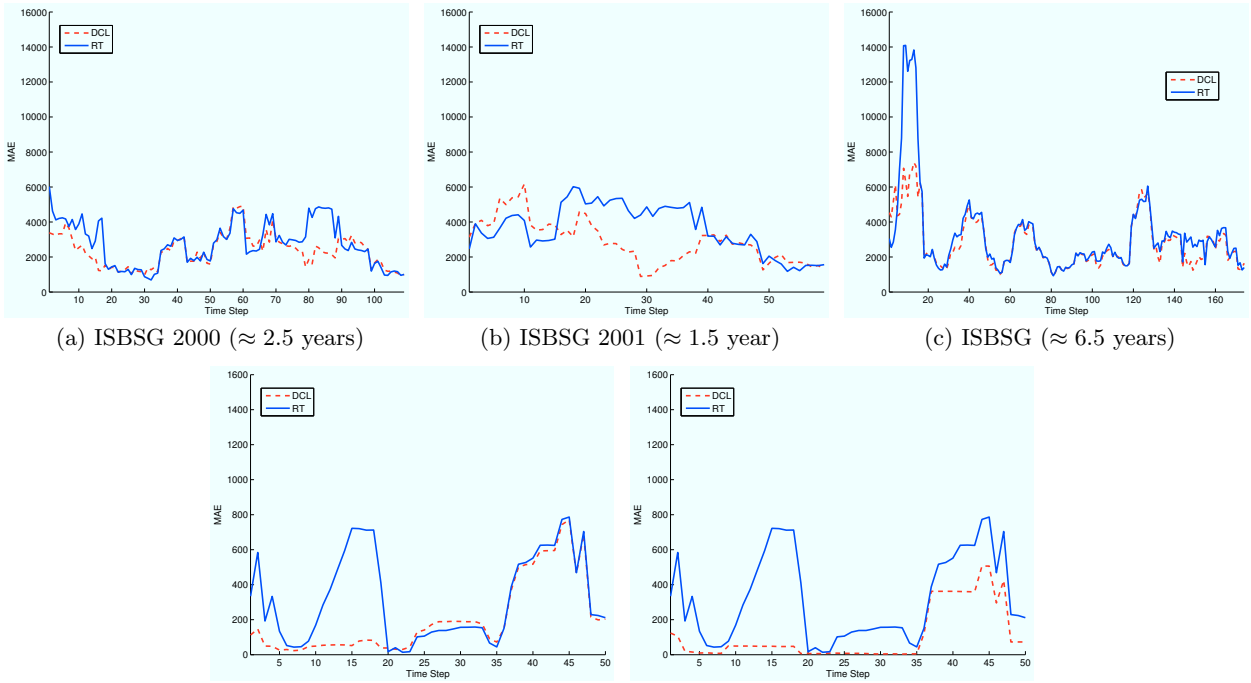
Approach	Rank Avg	Rank Std	z	P-value
DCL	1	0	3.4293	0.0006
CC-DWM	2	0	2.2045	0.0275
WC-DWM	3.2	0.45	0.7348	0.4624
RT	3.8	0.45	–	–

Table 4: Effect size Δ of the difference in overall MAE for each approach, using RT as the control approach. Effect sizes that could be considered as medium/high are in yellow/red (light/dark grey).

Data Set	WC-DWM	CC-DWM	DCL
ISBSG2000	0.0868	0.1490	0.3187
ISBSG2001	0.2501	0.5028	0.5474
ISBSG	0.0376	0.0876	0.1811
CocNasaCoc81	0.0768	0.2978	0.4541
CocNasaCoc81Nasa93	0.0768	0.6014	0.8378

7.2.3 *Performance at Each Time Step*

In addition to the overall performance across time steps, when working with online learning, it is also important to verify the performance at each time step. This allows checking whether a certain approach is better at some time steps, but worse at others. It also allows us to verify how much of



(a) ISBSG 2000 (≈ 2.5 years) (b) ISBSG 2001 (≈ 1.5 year) (c) ISBSG (≈ 6.5 years)

(d) CocNasaCoc81 (≈ 17 years) (e) CocNasaCoc81Nasa93 (≈ 17 years)

Figure 3: Performance of RT and DCL for each data set in terms of MAE. The number of years represents the period covered by the time steps considering the implementation date of the single-company projects.

CC data's potential benefit (section 5) is used by DCL.

Figure 3 shows the MAE at each time step for DCL and our control approach RT. There are several periods of around 20 time steps in which DCL outperforms RT. This number of time steps possibly involves several months (or even years) of worse RT estimations, which could have harmful consequences for a company. So, the improvements provided by DCL are considerable in terms of number of time steps.

By comparing figure 3 to 2, we can see that DCL managed to use the potential benefit from CC data in several cases. However, even though DCL rarely obtained worse performance than RT throughout the time steps, it still has room for improvement. Its MAE was a bit worse during the first 15 time steps for ISBSG2001. The reason for that may be related to the fact that two base models achieved similar performance during this period, but only one of them could be considered as the winner. Allowing more than one winner might improve the results in this case. The full potential benefit from the CC subsets Coc81 2 and Nasa93 was not used in the last 15 time steps for CocNasaCoc81 and CocNasaCoc81Nasa93, respectively, either. The reason for that is the filtering of CC models, which prevented using them in some cases where they were helpful. Even though DCL's rules still have room for improvement, they are more beneficial for SEE than DUR's, as explained in section 7.2.4.

7.2.4 DCL and DUR

Analyses of performance show that both DCL's new weight update rule and restricted use of CC models are key for DCL to improve on DUR's performance for ISBSG2000 and ISBSG2001. The difference in performance of the two versions was not considerable for ISBSG. For CocNasaCoc81, the restricted use of CC data prevents DCL of making full use of the CC models' potential. This causes DCL to obtain just slightly higher MAE than DUR, even though the new weight update rule on its own would have provided lower MAE. The restricted use of CC only considerably badly affected DCL in comparison to DUR for CocNasaCoc81Nasa93. However, it is worth noting that Nasa93 is more useful for predicting Cocomo Nasa than usual CC data sets are likely to be, as they both contain Nasa projects. So, we consider it safer to keep restricting the use of CC models in practice.

8. VALIDITY AND IMPLICATIONS

In order to ensure internal validity [20], we considered several parameter values for WC-DWM and CC-DWM, showing that their worse behaviour was not due to a bad parameters choice. RT's parameters were the most likely to produce good results in [18]. The construct validity was first dealt with by using MAE in section 5. This measure is not biased towards under or overestimations, being adequate for revealing the potential benefit of CC data. DCL was then compared against other approaches based on MAE, SA and Δ. So, we considered not only the performance, but also the magnitude of the differences in performance and their effect size. Friedman and post-hoc tests [5] were also used to show the significance of the differences in MAE. Besides never using a WC project for training before using it for testing, we considered five data sets to handle external validity. Three data sets with known WC chronological order were used both for revealing concept drift features in SEE and for evaluating approaches. Even though the chronological order is not known for the other two, they can still be used to evaluate whether the approaches are able to make use of CC data when/if they are beneficial, contributing to the generalisation of our results.

Our results show that organisations can use CC data to

improve SEEs provided by models that use solely WC data. They can acquire CC projects from different sources, e.g., ISBSG. Organisations willing to use an online learning tool based on DCL would need to collect a few attributes for each of their completed projects (e.g., development type, language type, development platform and functional size). These must be attributes that are also provided by the CC data set(s) and that can be used as input for DCL's CC models. After the completion of each project, these attributes together with the effort should be provided as a new training project to DCL, so that it can provide up-to-date estimations. It is worth emphasizing that DCL's potential may go well beyond SEE. Its use to other applications such as software defect prediction should be investigated.

9. CONCLUSIONS

This paper performs an analysis of the potential of CC data in comparison to WC data and presents a new approach to make use of this potential to improve SEE. It provides answers to the research questions as follows:

RQ1: Are CC data potentially beneficial for improving SEE? Under what conditions? Yes. CC models based on different subsets of CC data can become more or less beneficial/detrimental in comparison to WC models under concept drifting conditions. This paper is the first attempt at revealing the particular features of concept drifts induced by SEE tasks. They involve concepts being active for short numbers of projects, which were found to make CC data potentially helpful. Moreover, separating CC data into subsets based on productivity has shown to be potentially more useful than a single set of CC projects.

RQ2: Can this potential benefit be used? How? Yes. A new approach called DCL is proposed. It successfully improves SEE based on CC data by using weights to automatically determine when CC models are more or less helpful than a WC model. Each CC model is trained on a subset of CC data, representing different concepts and helping to deal with concept drifts in SEE. While DCL was successful in using CC data to improve SEE performance, an existing ML approach for handling drifts was not enough to improve SEE over a WC model not prepared for drifts.

Our results reveal that the online changing nature of software prediction tasks should be exploited, being an important issue to be considered in the next research frontier on software project estimation. For instance, DCL should be extrapolated and investigated in other software engineering applications, such as software defect prediction. Other future work includes testing additional weight update rules and approaches to filter CC models out; considering streams of incoming CC data and different numbers of estimation requests at each time step; and comparing DCL to other concept drift methods.

Acknowledgments: Work supported by EPSRC grant EP/J017515/1 and European Commission FP7 grant 270428.

10. REFERENCES

[1] M. Baena-García, J. Del Campo-Ávila, R. Fidalgo, and A. Bifet. Early drift detection method. In *IWKDDS*, pages 77–86, Berlin, Germany, 2006.

[2] B. Boehm. *Software Engineering Economics*. Prentice-Hall, Englewood Cliffs, NJ, 1981.

[3] M. Cartwright, M. Shepperd, and Q. Song. Dealing with missing software project data. In *METRICS*, pages 154–165, Sydney, 2003.

[4] S. Conte, H. Dunsmore, and V. Shen. *Software Engineering Metrics and Models*. Benjamin Cummings Publishing, Menlo Park, CA, 1986.

[5] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *JMLR*, 7:1–30, 2006.

[6] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: An update. *SIGKDD Explorations*, 11(1):10–18, 2009.

[7] B. Kitchenham, E. Mendes, and G. Travassos. Cross versus within-company cost estimation studies: A systematic review. *IEEE TSE*, 33(5):316–329, 2007.

[8] E. Kocaguneli, G. Gay, T. Menzies, Y. Yang, and J. W. Keung. When to use data from other projects for effort estimation. In *ASE*, pages 321–324, Antwerp, Belgium, 2010.

[9] J. Z. Kolter and M. A. Maloof. Using additive expert ensembles to cope with concept drift. In *ACM ICML*, pages 449–456, Bonn, Germany, 2005.

[10] J. Z. Kolter and M. A. Maloof. Dynamic weighted majority: An ensemble method for drifting concepts. *JMLR*, 8:2755–2790, 2007.

[11] C. Lokan and E. Mendes. Investigating the use of chronological splitting to compare software cross-company and single-company effort predictions. In *EASE*, page 10p, Bari, Italy, 2008.

[12] C. Lokan and E. Mendes. Applying moving windows to software effort estimation. In *ESEM*, pages 111–122, Lake Buena Vista, Florida, USA, 2009.

[13] C. Lokan and E. Mendes. Investigating the use of chronological split for software effort estimation. *IET-Software*, 3(5):422–434, 2009.

[14] C. Lokan and E. Mendes. Using chronological splitting to compare cross- and single-company effort models: Further investigation. In *ACSC*, pages 35–42, Wellington, New Zealand, 2009.

[15] E. Mendes and C. Lokan. Investigating the use of chronological splitting to compare software cross-company and single-company effort predictions: a replicated study. In *EASE*, page 10p, Durham, 2009.

[16] L. Minku and X. Yao. Using unreliable data for creating more reliable online learners. In *IJCNN*, pages 2492–2499, Brisbane, Australia, 2012.

[17] L. L. Minku, A. White, and X. Yao. The impact of diversity on on-line ensemble learning in the presence of concept drift. *IEEE TKDE*, 22(5):730–742, 2010.

[18] L. L. Minku and X. Yao. A principled evaluation of ensembles of learning machines for software effort estimation. In *PROMISE*, pages 10p, doi: 10.1145/2020390.2020399, Banff, Canada, 2011.

[19] L. L. Minku and X. Yao. DDD: A new ensemble approach for dealing with concept drift. *IEEE TKDE*, 24(4):619–633, 2012.

[20] M. L. Mitchell and J. M. Jolley. *Research Design Explained*. Cengage Learning, USA, 7th edition, 2010.

[21] S. Muthukrishnan. *Data Streams: algorithms and applications*. Now Publishers Inc., Hanover, MA, 2005.

[22] M. Shepperd and S. McDonell. Evaluating prediction systems in software project estimation. *IST*, 54(8):820–827, 2012.