Noname manuscript No.
(will be inserted by the editor)

Which Models of the Past Are Relevant to the Present?

A software effort estimation approach to exploiting useful past models*

Leandro L. Minku · Xin Yao

Abstract Background: Software Effort Estimation (SEE) models can be used for decision-support by software managers to determine the effort required to develop a software project. They are created based on data describing projects completed in the past. Such data could include past projects from within the company that we are interested in (WC projects) and/or from other companies (cross-company, i.e., CC projects). In particular, the use of CC data has been investigated in an attempt to overcome limitations caused by the typically small size of WC datasets. However, software companies operate in non-stationary environments, where changes may affect the typical effort required to develop software projects. Our previous work showed that both WC and CC models of the past can become more or less useful over time, i.e., they can sometimes be helpful and sometimes misleading. **Aims:** We aim at investigating how to automatically find if and when a model created based on past data represents well the current projects being estimated. **Method:** We propose an approach called Dynamic Cross-company Learning (DCL) to dynamically identify which WC or CC past models are most useful for making predictions to a given company at the present. DCL automatically emphasises the predictions given by these models in order to improve predictive performance. We compare DCL against existing WC and CC approaches and thoroughly analyse its behaviour. **Results:** DCL is successful in improving SEE by emphasizing the most useful past models. Our detailed analysis of DCL's behaviour strengthens its external validity.

Keywords Model-based software effort estimation · machine learning · cross-company learning · online learning · non-stationary environments.

* A preliminary version of this work appeared as [43].

Leandro L. Minku
Department of Computer Science, University of Leicester, University Road, Leicester, LE1 7RH, UK. Email: leandro.minku@leicester.ac.uk

Xin Yao
CERCIA, School of Computer Science, The University of Birmingham, Edgbaston, Birmingham, B15 2TT, UK. Email: x.yao@cs.bham.ac.uk

List of Abbreviations and Specialist Terms

AddExp	Additive Expert Ensemble machine learning approach.
Bagging	Bootstrap Aggregating machine learning approach.
Base learners	Machine learning models composing an ensemble and their corresponding learning algorithms.
CC	Cross-Company; other companies. In this paper, we will use the term CC loosely. For example, projects from different departments within the same company could be considered as CC projects if such departments employ largely different practices.
Data stream	Sequence of chronologically ordered examples.
DCL	Dynamic Cross-Company Learning approach proposed in this work.
DCL-F	DCL using only filtering.
DCL-N	DCL using no dynamic weighting and no filtering.
DCL-W	DCL using only dynamic weighting.
DDD	Diversity for Dealing with Drifts machine learning approach.
Δ	Glass' effect size.
DWM	Dynamic Weight Majority machine learning approach.
EBA	Estimation By Analogy.
Ensemble	Set of machine learning models grouped together with the aim of improving predictive performance.
Functional size	Software size measurement based on the amount of functionality to be delivered by the software.
Input attribute	Independent variable; feature describing a project / example.
ISBSG	International Software Benchmarking Standards Group.
k-NN	k-Nearest Neighbours machine learning approach.
MAE	Mean Absolute Error measure of predictive performance.
ML	Machine Learning.
MLP	MultiLayer Perceptron.
Output attribute	Dependent variable; feature that the machine learning approaches aim to predict. In this work, this is the effort.
Predictive performance	Measure of how good the predictions / estimations are; accuracy of predictions / estimations.
RBF network	Radial Basis Function network; machine learning approach.
Relevancy filtering	Approach that eliminates CC projects that are too different from the WC projects being predicted.
Rguess	Random guess.
RT	Regression Tree machine learning approach.
SA	Standardised Accuracy measure of predictive performance.
SEE	Software Effort Estimation.
Time step	Moment in time when a new training example is received.
Training example	In this work, this is a project for which both the input and output attributes are known. It can be used for training / updating machine learning models.
WC	Within-Company.

1 Introduction

Software Effort Estimation (SEE) is the process of estimating the effort required to develop a software project. Software effort is typically the main cost driver in software projects [22, 59]. Both over and underestimations of effort can cause problems to a company. For instance, overestimations may result in a company losing contracts or wasting resources, whereas underestimations may result in poor quality, delayed or unfinished software projects.

SEE is a difficult task. Human-made effort estimations may be strongly affected by effort-irrelevant and misleading information, such as the font or margin size of specifications [21]. Sometimes, software engineers may not improve their effort estimations even after feedback about their estimates is provided [18]. Therefore, several Machine Learning (ML) approaches have been investigated to automatically build SEE models [62]. However, SEE models created using ML might not capture some human factors that influence the effort required to develop software projects. Therefore, SEE models should not be used as replacements for experts and experts should not blindly trust such models. Instead, we believe that SEE models should be used as decision-support tools to help experts to perform or re-think their estimations.

For example, if an expert estimation is similar to the estimation given by a model, then we have an increased confidence on the estimate. If the two estimations differ considerably, then the expert can analyse the project further to gain extra insights into what effort best reflects the project. Depending on his/her further analysis, he/she may decide to keep his/her own estimate, or adopt the model's estimate, or an estimate in between the two estimates. In short, we envision learned models to work in conjunction with human experts.

ML approaches create SEE models based on data describing projects completed in the past. Such data could include past projects from within the company that we are interested in (WC projects) and/or from other companies (CC projects). It has been suggested that early SEE models such as COCOMO and SLIM need to be calibrated to the context of specific companies based on data from within these companies in order to work effectively [11, 25]. Other authors have also suggested that SEE models should be built based on WC data [29, 16]. However, companies may face difficulties in terms of WC data collection, e.g. [23], (1) the time required to accumulate enough data [14] on past projects from a single company may be prohibitive; (2) by the time the dataset is large enough [14], technologies used by the company may have changed, and older projects may no longer be representative of current practices; and (3) care is necessary as data need to be collected in a consistent manner.

In an attempt to overcome these problems, the use of CC models for SEE has been investigated [23, 26, 42]. CC models are typically defined as those built using datasets containing data from several companies. Several CC datasets are available for SEE (http://openscience.us/repo/) and there are even organisations worldwide that use large proprietary CC datasets with tool support to provide estimation and benchmarking services. An example is the Interna-

tional Software Benchmarking Standards Group (ISBSG) [20], which provides tools to estimate effort and benchmark productivity using their CC data. ISBSG also sells their CC data to those companies that wish to use their own tools for estimation and benchmarking purposes.

However, studies comparing CC and WC SEE models suggested that CC models typically perform similar or worse (and no better) than WC models [23]. A more recent study [43] further confirmed that CC models can indeed sometimes perform worse than WC models. Nevertheless, this study also revealed that CC models have the potential to outperform WC models *depending on the time period analysed* [43]. This finding is in accordance with the fact that companies operate in non-stationary environments. 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. Such changes can affect the predictive performance of SEE models. In fact, both WC and CC models of the past can become more or less useful over time, i.e., they can sometimes be helpful and sometimes misleading [43]. SEE models developed at a certain point in time may become obsolete. In a similar way, models that were good in the past and poor at present may become useful again in the future [43], as a company may start behaving similarly to a previous situation.

It is likely that whether or not CC data are useful as training data depends on how similar they are to the current target projects. If the CC projects are similar to the current target projects, they are likely to be useful for these projects. Otherwise, they are not currently directly useful. It is also important to note that, even though the terms CC and WC are widely used in software effort estimation, WC projects can be themselves dissimilar / heterogeneous. Therefore, the terms CC and WC are not ideal. In this paper, we will use the term CC loosely. For example, projects from different departments within the same company could be considered as CC projects if such departments employ largely different practices.

If one can successfully identify which CC and WC models are currently the most useful ones, it may be possible to emphasise the right models to improve SEE. With that in mind, this paper aims at answering the following research questions:

RQ1 How can we know which model from the past best represents the current projects being estimated?
RQ2 Can that information help improving SEE?

In order to answer these research questions, we propose an approach called Dynamic Cross-company Learning (DCL) able to identify the models that reflect well the current projects being estimated automatically. It then emphasises these models in order to improve SEE.

Our proposed approach was preliminarily presented in [43]. Its mechanism to emphasize the best models has also been adopted by a later approach [45]. However, those works did not investigate whether the mechanism to emphasize

the right models really works as expected and why. They did not validate the improvement in predictive performance that can be achieved by emphasising such models in comparison to an approach that always gives the same emphasis to all models either. Even though DCL was compared against a WC model and a new approach for dealing with non-stationary environments from the ML literature, typical CC SEE approaches were not included in the analysis [43]. Since that study, a new competitive CC SEE approach has also been proposed [61]. So, ideally DCL should be compared against that approach. Furthermore, no previous work provided a thorough understanding of DCL's behaviours. A thorough understanding is important when proposing new approaches, as it can strengthen its external validity by identifying the situations where the approach is successful and the situations where it could fail. It can also give an insight into what components of an approach make it successful and whether all of them are really necessary; and into how robust the approach is to different parameter choices and types of base learner. The current work performs new analyses addressing all these issues (sections 6 to 10), providing a thorough validation of DCL and its ability to emphasise the right models.

The rest of this paper is organised as follows. Section 2 presents related work. Section 3 presents our formulation of the problem. Section 4 presents the proposed approach Dynamic Cross-company Learning (DCL). The approach was preliminarily presented in [43] and answers part of RQ1. Section 5 describes the datasets used in our study. Section 6 provides an analysis of the ability of DCL to emphasize the right models and a detailed understanding of why and when DCL is expected to succeed or fail to emphasize the right models. It investigates DCL's ability to answer RQ1. Section 7 validates DCL against an approach that always gives the same emphasis to all existing SEE models, i.e., it evaluates the improvement in SEE predictive performance achieved by emphasising the right models. It also analyses which of the two main components of DCL is key for its improved predictive performance. Section 8 compares DCL against a corresponding WC model and other WC and CC approaches. Together, sections 7 and 8 answer RQ2. Section 9 presents a study showing that DCL is robust against the type of model used in combination with it. The study also reveals what types of model are likely to do better in combination with DCL. Section 10 provides an analysis of DCL's sensitivity to parameters. These sections contribute to the external validity of our study. Section 11 discusses threats to validity. Section 12 presents the conclusions, implications to practice and future work.

2 Related Work

2.1 ML for SEE Assuming No Chronology

There has been much work on SEE in the software engineering literature [22, 23]. Algorithmic SEE models have been studied for many years [11, 22]. Among them, ML algorithms have been increasingly investigated as automated SEE

approaches [22]. Most work using ML for SEE implicitly assumes that the projects used to build predictive models have no temporal order / chronology. Such work does not ensure that the software projects used to build a ML model are projects completed before the projects used for testing this model. This means that models used for making effort estimations are potentially trained with projects that would not have been available for training in a real world scenario. This section explains some work that assumes no chronology.

An important work in this area is that of Shepperd and Schofield [57], who used a k-Nearest Neighbour (k-NN) algorithm [10] based on normalised attributes and Euclidean distance as the similarity measure. This approach is also known as estimation by analogy. Despite being first used for SEE more than fifteen years ago, this approach has been shown to be able to achieve competitive results in comparison to recent techniques, depending on the dataset [44].

More recent work has been emphasising the relatively good predictive performance achieved by ensembles of learning machines [32, 44, 27] and local methods that make estimations based on completed projects similar to the project being estimated [44, 42, 8]. For instance, Regression Trees (RTs), Bagging ensembles of MultiLayer Perceptrons (Bag+MLPs) and Bagging ensembles of RTs (Bag+RTs) have been shown to perform well across several datasets [44].

Several SEE studies have also tried to use CC models in an attempt to deal with the fact that WC training sets are frequently not large enough to represent the whole population of projects well, resulting in poor SEE models. For example, some studies used CC training examples to augment their existing WC training sets [33]. The resulting augmented training set was then used to build models to make predictions in the WC context. Others used solely CC training examples to build such models [12, 63]. Both the former and latter types of studies have found that the resulting models obtained similar or worse predictive performance than models trained solely on WC training examples [23].

More recent work on relevancy filtering has demonstrated that eliminating CC projects that are too different from the projects being predicted can more frequently lead to CC models able to achieve similar predictive performance to WC models [61, 28]. However, measuring similarity between training and target projects coming from different and potentially heterogeneous sources is not straightforward. This is because similarity here involves not only the space of available input attributes, but also the output attribute (effort). Two projects that are similar in the input space may still be different in the output space if they come from heterogeneous sources. This can happen, for example, as a result of different interpretations given to the levels of subjective input attributes during data collection, as a result of the unavailability of certain relevant input attributes, or as a result of different definitions of work hours (e.g., based on unpaid overtime). Therefore, measuring similarity based solely on the input attributes will not always work well. This is a potential reason why

CC SEE approaches based on similarity on the input space such as Relevancy Filtering [61, 28] sometimes perform worse than WC models.

An insightful work [42] in the context of predicting software effort and defect proneness is based on clustering WC+CC examples, and then creating prediction rules for each cluster. These prediction rules are aimed at finding features that lead to less effort or fewer defects. Given a certain cluster, its neighbouring cluster with the lowest required efforts/defects was referred to as the envied cluster. When making predictions for WC projects from a cluster, rules created using only the CC examples from the envied cluster were better than rules created using only the WC examples from the envied cluster. So, the authors recommended to cluster WC+CC examples, but to learn rules using solely the CC examples from the envied cluster. These results are very encouraging and motivate further investigation of CC SEE, as they suggest that CC models may also be able to achieve better predictive performance than WC models in SEE.

2.2 Chronology-Based ML for SEE

The approaches described above do not consider the chronology of the software projects being used for creating and evaluating the SEE models. However, SEE operates in online learning scenarios where new completed projects arrive over time following a temporal order. Such scenarios are 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. For instance, Kitchenham et al. [24] reported that the best fitting regression model changed substantially over time in a case study with a WC dataset. Premraj et al. [53] also reported that productivity changed over time in a study with a CC dataset. In order to reflect a real SEE scenario more closely, the chronology of projects should ideally be considered when developing/evaluating SEE models.

Existing approaches involving chronology-based ML for SEE can be divided into three types: chronological splitting (section 2.2.1), moving window (section 2.2.2) and time transfer (section 2.2.3).

2.2.1 Chronological Splitting Approaches

ML evaluation procedures for stationary environments randomly split data into training and testing data without taking chronology into account. The training data are used to create predictive models, whereas test data are used to estimate the models' predictive performances on unseen data. Examples of such procedures are repeated (random) holdout and cross-validation. These procedures are unsuitable for evaluating predictive models in non-stationary

environments, because changes may cause examples from different periods of time to have different characteristics. For instance, consider that the productivity of a company significantly changed over time, affecting the relationship between available input attributes and effort in a company. If we allow a ML model to be trained using projects from the future to predict projects from the past, this model will use examples that reflect the new relationship between input attributes and effort, which would not have been available in a real world scenario. As a result, the testing performance obtained by such models will not reflect the predictive performance that would be achieved in practice.

In order to account for that, chronological splitting approaches create data splits ensuring that all training projects have been completed before the starting date of the testing projects. For example, Lefley and Shepperd [33] performed a SEE study to compare the predictive performance of genetic programming against neural networks, k-nearest neighbours and least squares regression. In order to perform this comparison in a more realistic manner, they used a date-based splitting of their dataset into two partitions. The first partition included 48 WC and 101 CC projects completed by 15th October 1991, and was used for training. The second partition included 15 WC projects that started after this date, and was used for testing. Sentas et al. [55] also adopted date-based splitting in two of the three datasets used to compare ordinal regression and stepwise linear regression.

Auer et al. [7,6] performed an analysis of input attribute weighting methods for analogy-based SEE. They explained that SEE datasets typically grow over time as companies take on new projects. Therefore, a realistic SEE procedure would be to (1) measure the input attributes of the project to be estimated, (2) estimate the effort for this project based on the existing SEE model, and (3) upon a project's completion, add its input attributes and effort to the dataset and re-build or re-calibrate the SEE model based on the new, larger, dataset. This approach, where the SEE model used to estimate a given project is trained on all previously completed projects, can be referred to as project-by-project splitting or growing portfolio. It allows us to investigate how SEE models change as they are updated over time. Auer and Biffl's work [6] was based on five WC datasets and Auer et al.'s work [7] was based on eleven WC datasets. However, it is not entirely clear if chronology was considered in all of them. Several later studies also involved comparisons with growing portfolio approaches [36,41,5,39,3,4].

In practice, we wish to estimate the effort for a given project soon after its commencement based on all completed projects available by that time. We may also wish to update this initial estimate if more completed projects become available before the end of this project. Therefore, whenever a new project is completed, one may wish to predict a given number of incomplete or future projects based on all completed projects. For example, consider that we have three projects p_t, p_{t+1} and p_{t+2}, completed at times t, $t+1$ and

$t + 2$, respectively [1]. At time t, we may wish to estimate projects p_{t+1} and p_{t+2} based on a model trained with all projects completed up to time t. Then, once we reach time $t + 1$, we may wish to provide an updated prediction for project p_{t+2} based on a model trained with all projects completed up to time $t + 1$. This approach has been used, for example, in a study of the impact of parameter tuning on SEE [58]. This study was based on three WC datasets and five ML approaches (MLP, Bag+MLP, RT, Bag+RT and k-NN). It shows that the best parameters to be used with ML approaches can change over time. This issue affects some ML approaches such as k-NN more than others such as Bag+RTs. Other studies also involved this type of chronological splitting, but in the context of time transfer approaches [49, 45].

Even though the studies above used chronological splitting, they did not investigate the impact of using chronological splitting in comparison with random splitting. With that in mind, Lokan and Mendes compared WC models created based on growing portfolio (WC1) with leave-one-out (WC2) and leave-two-out (WC3) cross-validation [35]. They also compared CC models created based on growing portfolio (CC1) with a CC model created based on the whole CC dataset (CC2). Their study was based on multivariate stepwise regression and one dataset. In terms of absolute errors, WC1 performed similarly to WC2 and WC3, and CC1 performed similarly to CC2. However, in terms of z values, WC1 was significantly worse than WC2 and WC3, and CC1 was significantly better than CC2. This demonstrates that the results obtained by using random splitting can differ from the results obtained using chronological splitting. A similar study in the context of data-based splitting demonstrates that date-based splitting can also sometimes lead to different results from random holdout [38]. MacDonell and Shepperd [41] also investigated growing portfolio in comparison with leave-one-out cross-validation based on least squares linear regression and a WC dataset. Their results suggest that these evaluation approaches lead to different results, even though their analysis is not based on statistical tests. Overall, these studies show that it is important to consider chronology in order to better reflect the SEE procedure used in practice.

2.2.2 Moving Window Approaches

Even though the studies presented in section 2.2.1 considered chronology, they did not consider that older projects may become obsolete and hinder the predictive performance if included in the training set. Kitchenham et al. [24] recommended to discard old projects from the training set based on a moving window approach as follows. For each new project p_t to be estimated, a SEE model should be created based on a "window" containing projects p_{t-1} to p_{t-n}. As this approach creates windows based on the number of previous projects to be included, it can be referred to as fixed-size window approach. In

[1] Please note that each time step (e.g., t, $t+1$ and $t+2$) refers to the completion time of a project, and not to the whole duration (from start to completion time) of this project.

Kitchenham et al.'s case study with the WC CSC dataset, $n = 30$ was used. This means that each window had a fixed size of 30 projects. This approach was proposed due to the observation that the best fitting regression model changed substantially over time in their case study.

Lokan and Mendes [36]'s work is to the best of our knowledge the first work to provide a detailed investigation of whether moving windows can improve predictive performance. Their work revealed that SEE models trained on fixed-size windows can provide significantly better predictive performance than the growing portfolio approach [36]. However, whether or not fixed-size windows were beneficial depended on the window size. In their case study using multivariate stepwise regression and the WC ISBSG dataset (a subset derived from ISBSG Release 10), smaller windows (from 20 to 40 projects) had a detrimental effect in comparison with growing portfolio. Larger windows (from 85 to 120 projects) had a positive effect on predictive performance in terms of magnitude of the relative error, but led to similar predictive performance in terms of absolute error.

Amasaki and Lokan [3] also compared fixed-size windows against growing portfolio as part of their study with the same WC ISBSG dataset as [36]. However, they used linear regression based on input attributes selected with Lasso [60] instead of stepwise regression. The reason for using Lasso was that their preliminary results showed that Lasso provided more accurate estimates. Fixed-size windows achieved similar absolute error to growing portfolio for window sizes from 20 to 40, and significantly better absolute error for window sizes between 40 (exclusive) and 120 (inclusive). In terms of magnitude of the relative error, fixed-size windows started to provide better predictive performance even for some window sizes smaller than 40.

MacDonell and Shepperd [41] investigated the predictive performance of fixed-size windows with size of five projects in a study based on least squares linear regression and the WC MacDonell dataset. Even though their analysis is not based on statistical tests, it suggests that moving windows can provide much better results than growing portfolio.

A further study [40] found fixed-size windows to obtain either similar or significantly worse predictive performance than growing portfolio when using multivariate stepwise linear regression for the WC Finnish dataset (a subset of the Finnish dataset).

All results above were based on linear regression, which can be considered as a global learning approach. Global learning approaches make estimations based on models representing the whole training set. Different from global learning approaches, local learning approaches make estimations based only on the training projects most similar to the project being estimated. Amasaki et al. [5] explained that, as the estimations are based on a subset of all training projects, moving window approaches might be less useful for local learning approaches. Therefore, they performed a comparison between fixed-size window and growing portfolio using the local approach k-NN and two WC datasets (CSC and Maxwell). Their experiments found no statistically significant differences in predictive performance between the two approaches.

A later study [1] with the WC ISBSG dataset showed that statistical significances were found in favour of fixed-size windows when using k-NN, even though for less window sizes than when using linear regression. Therefore, fixed-size windows were effective both with linear regression and k-NN for this dataset when compared against growing portfolios. However, the degree of effectiveness was higher when using linear regression. A very recent study [4] with the WC Finnish dataset showed that fixed-size windows obtained either similar or significantly worse predictive performance than growing portfolios when using k-NN.

Another type of moving window approach defines window size in terms of duration as follows. For each new project to be estimated, a SEE model should be created based on a window containing all projects whose development span occurred during the last n months. This means that the number of projects within the window is not fixed – it depends on the number of projects developed during the last n months. The advantage of duration-based windows is that they allow for a clear cut in terms of how recent projects must be in order to be included in the window, ensuring that projects deemed old are not included. The disadvantage is that there is no explicit control over the number of projects included in the window. If the number of projects developed during the last n months is small, the small training set used to create the SEE models may result in poor predictive performance.

Lokan and Mendes [40] compared duration-based windows against growing portfolio and fixed-size windows, based on stepwise multivariate regression and two WC datasets (WC ISBSG and WC Finnish). For the WC ISBSG dataset, their analysis shows that duration-based moving windows can be beneficial to predictive performance in comparison with growing portfolio, depending on the duration. Duration of around 36 months provided the most promising results for their dataset. When using the most promising duration and number of projects for duration-based and fixed-size windows, duration-based moving windows performed statistically similarly to fixed-size windows. However, for the WC Finnish dataset, both duration-based windows and fixed-size windows obtained either similar or significantly worse predictive performance than growing portfolio.

Amasaki and Lokan [4] further investigated the predictive performance of duration-based windows when using k-NN. They also used the WC Finnish dataset. However, they found that duration-based windows achieved either similar or better predictive performance than growing portfolio, depending on the duration. The results when using k-NN with duration-based windows are thus very different from the results using k-NN with fixed-size windows and from the results using linear regression with duration-based windows.

Amasaki and Lokan also proposed the use of weighted fixed-size [3] and weighted duration-based [2] windows. Weighted windows give more weight to more recent projects within the window than to older projects. Weighted and unweighted windows were compared using linear regression with Lasso input attributes selection and the WC ISBSG dataset. The analysis shows that weighted fixed-size windows significantly improved predictive performance in

larger windows, and significantly worsened predictive performance in smaller windows [3]. Weighted duration-based windows improved predictive performance significantly especially for larger windows. For some other window sizes, weighted duration-based windows were detrimental [2].

All the studies above were based on WC datasets. The fact that windows can sometimes be beneficial is an indication that the (unknown) real underlying function mapping input attributes to effort in a single company may change over time, emphasising the importance of proposing and investigating approaches able to deal with such changes at the same time as not hindering predictive performance when there are no changes. The main difficulty with using moving windows is that their predictive performance is highly dependent on the window size and they may be detrimental to some datasets. The studies explained in this section show that discarding old data can be helpful, but is not always advisable.

2.2.3 Time Transfer Approaches

Our work [43] showed that CC and WC models can become more or less useful over time, sometimes representing well the current projects being estimated and sometimes being misleading. These results corroborate the finding that discarding old data is not always advisable [40]. Based on that, a dynamic adaptive approach (DCL) was proposed to automatically adapt to changes by making use of old data when they are helpful. This type of approach can be seen as transferring knowledge from different periods of time to the present when they are beneficial.

DCL is to the best of our knowledge the first approach able to use CC data for improving predictive performance over WC SEE models [43]. It is able to achieve that by identifying which models among CC and WC models created with past data are most useful to a company at each given point in time based on a weighting mechanism. Its weighting mechanism to emphasize the best models has also been adopted by a later approach [45]. However, as explained in section 1, DCL and its weighting scheme have not been thoroughly evaluated yet. DCL is explained in detail in section 4.

2.3 ML Literature on Predictive Models for Non-Stationary Environments

The ML literature contains several approaches designed for dealing with online and non-stationary environments. However, such approaches typically assume the availability of large amounts of data, forming a data stream [52] unlikely to exist in SEE. A widely known approach for classification tasks in this type of environment is Dynamic Weight Majority (DWM) [31]. DWM creates different base learners[2], each associated with a dynamic weight which is reduced when the base learner gives a wrong prediction. Base learners are dynamically added

[2] The models (and their corresponding learning algorithms) composing an ensemble are referred to as its *base learners*.

and removed, facilitating adaptation to changes. DWM's predictions are based on the weighted majority vote among the base learners.

Additive Expert Ensemble (AddExp) [30] 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 less robust to noise. Another approach is Diversity for Dealing with Drifts (DDD) [49]. Its main idea is that very highly diverse ensembles (whose base learners produce very different predictions from each other) are likely to present poor predictive performance under stable conditions, but may become useful when there are changes. So, DDD maintains both a low diversity ensemble and a high diversity ensemble, which is only activated upon change detection.

3 Formulation of the Problem

We formulate SEE as an online learning problem in which a new *completed* project implemented by a single company is received as a *training example* at each *time step*, forming a WC data stream. These training examples can be used to create a WC SEE model. Different from typical online data stream problems [52], even though new projects arrive with time, the volume of incoming training data is small. So, there are no tight space or time constraints. For instance, it is acceptable for a new SEE model to be created from scratch whenever a new training project becomes available.

At each time step, we wish to determine which SEE models among a set of WC and CC models are currently most useful for the single company, i.e., which SEE models best represent the current relationship between input and output attributes for the projects being estimated. We also wish to predict the effort of a given number of future projects from the WC data stream, i.e., projects that have not yet completed and whose actual effort is still unknown. We consider ten as a reasonable number of future projects to be predicted in this study, although our approach is applicable to any value.

It is important to emphasize that only *past completed projects* with known effort (training examples) can be used for determining which SEE models are currently most useful and for training / updating existing models.

4 Dynamic Cross-company Learning (DCL)

This section describes an approach called Dynamic Cross-company Learning (DCL), which is able to automatically identify which existing CC or WC models are currently the most relevant for making SEEs for a company (RQ1). These models are then emphasized in order to improve SEE for this company (RQ2).

The idea behind DCL is that models performing poorly at a certain moment may become beneficial in the event of changes in the environment [47]. For instance, a CC model may perform poorly for some period of time and perform

relatively well in a later period [43]. So, DCL maintains a memory of m models trained on past CC data and one model specific for WC data stream learning. Each model is associated with a weight representing how useful it currently is, inspired by Kolter and Maloof's ML approaches [31,30]. These weights are dynamically updated and allow DCL to emphasize estimations given by CC data models when they are useful. They are proposed to address research question RQ1 outlined in section 1. DCL's estimations are then based on the weighted average of the base learners' estimations. In this way, we expect DCL to improve SEE by giving different emphasis to different models (RQ2).

As a backup measure, should the weighting mechanism fail at some point, DCL also restricts the use of CC models. 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 to the weighted average for estimating this project. Filtering out CC data for localization has been shown to be useful for SEE [61,28].

The method to filter CC models out in DCL is as follows. An impactful input attribute is used to define whether a certain WC project to be predicted is "too different" from the projects used to build a CC model. For example, for all the WC datasets used in this study, size is likely to be the most impactful input attribute, given that regression trees trained on these datasets selected size as the top-level attribute. A CC model is then considered in the estimation of 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 very small number of WC training projects, filtering of CC models is only applied after a considerable number T_{start} of WC training projects have been presented, where T_{start} is a pre-defined parameter. It is worth noting that other input attributes than size should be used for filtering if they are deemed more influential than size. This may happen, for example, when projects have a high level of reuse and integration.

The CC models can be any CC models available to the company that we are interested in. In particular, if the company has access to the CC training projects themselves, m different CC training sets can be created based on some a priori knowledge. For example, different sets can be created for different companies, or different sets can be created by grouping together CC projects with similar productivity. It is worth noting that we use the term CC loosely here. For example, projects from different departments within the same company could be considered as CC projects if such departments employ largely different practices. Separation based on productivity ranges can be particularly useful and easily automated. The reason for the usefulness of productivity-based separation is that different productivity ranges can simulate different possible situations of a company. When a change happens, a company may become more or less productive, and CC models built using different produc-

Algorithm 1 DCL

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

1: **for** each CC model $L_c, 1 \leq c \leq m$ **do**
2: $w_c = 1/m$ {Initialise weight for CC models.}
3: $w_{m+1} = 0$ {WC model weight.}
4: Allow using DCL for predictions while there is no WC training project available
5: **for** each new WC training project $(\mathbf{x}, y)$ **do**
6: $winner = \text{argmin}_{i,1 \leq i \leq m+1} |L_i(\mathbf{x}) - y|$
7: **for** $loser, 1 \leq loser \leq m+1 \wedge loser \neq winner$ **do**
8: $w_{loser} = \beta w_{loser}$, where
$\beta = \beta_c$ if $loser <= m$ and $\beta = \beta_w$ otherwise.
9: **if** $(\mathbf{x}, y)$ is the first WC training project **then**
10: $w_{m+1} = \frac{1}{m+1}$
11: Divide each weight by the sum of all weights
12: Use $(\mathbf{x}, y)$ to learn WC model L_{m+1}
13: Allow using DCL for predictions while no new WC training project is made available

tivity ranges could become more or less beneficial. Note that productivity is based on effort. So, we cannot use the productivity of the project being estimated to decide which CC models are likely to be more beneficial for this project. Instead, DCL determines which base learners are likely to be more or less beneficial at each time step based on dynamic weights updated through its learning algorithm whenever a new WC project is completed.

Algorithm 1 presents DCL. Existing CC models can be learnt beforehand and provided to the algorithm as input arguments. DCL initialises the weights associated to each CC model with the value $1/m$ (line 2). Weights associated with WC models are initialized to zero (line 3), so that DCL can be used for predictions before any WC training project becomes available, i.e., based solely on CC models (line 4).

After that, whenever a new WC training project is made available, it is used to (1) update the weights used to determine which SEE models are most relevant at present and (2) train the WC model. One WC training project from the stream is received at each iteration, which corresponds to one time step. The weight update rule is shown in lines 6–11. Each base learner 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 6). This can be either the WC or a CC model. The weights associated with all the loser models are 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. If the current WC training example is the first WC training example, the weight of the WC model is changed from 0 (zero) to $1/(m+1)$ (line 10). After all weights are updated, they are divided by the sum of all weights (line 11).

The WC model's training is done at the end of the iteration (line 12). 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. DCL is made available for predictions after the WC model is trained at the end of each iteration (line 13) and until a new WC training project arrives. Typically, one would like to predict a certain number of future projects from the WC data stream before a new WC training project arrives.

5 Datasets

Five different datasets were used in our study: ISBSG2000, ISBSG2001, ISBSG, Nasa60Coc81 and Nasa60Coc81Nasa93. These include both datasets derived from the International Software Benchmarking Standards Group (ISBSG) Repository [20] and the PRedictOr Models In Software Engineering Software (PROMISE) Repository [54]. Each dataset is composed of a WC data stream, and a number of CC subsets.

5.1 ISBSG Datasets

Three SEE datasets were derived from ISBSG Release 10, which contains software project information from several companies. Information on projects belonging to a single company was provided to us upon request. 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 datasets were then created:

- 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 dataset can be used to simulate the case in which it is known that other companies can be more evolved than the single company analysed.

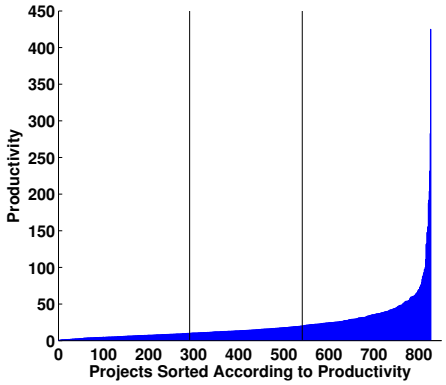


Fig. 1 Sorted productivity of the 826 ISBSG CC projects. The vertical black lines separate the projects into three subsets according to the ranges in table 1.

The split date has been chosen so as to provide increments of roughly the same size. ISBSG2001 contains a few more than 60 WC projects, ISBSG2000 contains almost $2 \cdot 60$ projects and ISBSG contains a few more than $3 \cdot 60$ projects. Even though these datasets are not entirely independent, they can demonstrate how well DCL is able to cope with different data stream lengths. Indeed as can be seen in section 6 (figures 3(a), 3(b) and 3(c)), the weights learned by DCL differ for these three datasets.

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 [13] imputation was used for dealing with missing attributes for each dataset separately.

The CC models used by DCL were trained on CC project subsets created according to their normalised level 1 productivity rate provided by the repository. The separation into subsets 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. Note that each of the three datasets derived from the ISBSG repository thus contain three different CC subsets and one WC data stream.

5.2 Nasa60Coc81 Dataset

Nasa60 [3] and Cocomo81 are two software effort estimation datasets available from the PROMISE Repository. Nasa60 contains 60 Nasa projects from 1980s-1990s and Cocomo81 consists of the 63 projects analysed by Boehm to develop

[3] Nasa60 has also been named Cocomo Nasa in the past, and is not available for download from the current PROMISE repository.

Table 1 Ranges of productivity for CC subsets. For ISBSG2000, ISBSG2001, ISBSG and Nasa60Coc81, these ranges were used to create different CC subsets. For Nasa60Coc81Nasa93, these ranges represent the different productivity values present in the original Cocomo81 and Nasa93 datasets.

CC Subset		Productivity Band	Number of Examples
ISBSG2000	CC-0	[0.7,5]	56
	CC-1	(5,13]	57
	CC-2	(13,155.7]	55
ISBSG2001	CC-0	[0.7,6]	72
	CC-1	(6,14]	79
	CC-2	(14,155.7]	73
ISBSG	CC-0	[0.3,10]	291
	CC-1	(10,20]	250
	CC-2	(20,424.9]	285
Nasa60Coc81	Coc81-0	[0.7,2.85]	19
	Coc81-1	(2.85,6.6]	20
	Coc81-2	(6.6,49]	24
Nasa60Coc81Nasa93	Coc81	[0.7,49]	63
	Nasa93	[0.59,89.38]	93

the software cost estimation model COCOMO [11]. Both datasets contain 16 input attributes (15 cost drivers [11] and number of lines of code) and one output attribute (software effort in person-months). Cocomo81 contains an additional input attribute (development type) not present in Nasa60, which was thus removed.

Nasa60's projects were considered as the WC data and Cocomo81's projects were considered as the CC data. There is no information on whether Nasa60's projects are sorted in chronological order. The original order of the Nasa60 projects was preserved in order to simulate the WC data stream. Even though this may not be the true chronological order, a simulated chronological order can be used to show (1) whether DCL would be able to identity which model is more relevant with that order of projects and (2) whether DCL can benefit from the CC models to improve SEE. As for Cocomo81, the dataset provided by the PROMISE repository is sorted according to project identifier. In this study, we have sorted Cocomo81's projects according to the completion year provided in Boehm's book [11].

In order to create different CC models for DCL, the productivity in terms of effort divided by the number of lines of code was calculated for Cocomo81. The productivity values are skewed, similarly to ISBSG's, shown in figure 1. CC projects were then separated into subsets according the ranges shown in table 1. Each CC subset was used to train one CC model. Note that Nasa60Coc81 is thus composed of three CC subsets (derived from Cocomo81) and one WC data stream (Nasa60).

5.3 Nasa60Coc81Nasa93 Dataset

This dataset is also composed of Nasa60 and Cocomo81, but it includes an additional dataset called Nasa93, which contains 93 Nasa projects from 1970s-

1980s and has the same input and output attributes as Cocomo81. As with Cocomo81, the attribute development type was removed in order to keep compatibility with Nasa60. Nasa60 and Nasa93 are both composed of Nasa's projects, and they have an overlap of 55 projects. Nasa60 is used here as the WC data stream, whereas Cocomo81 and Nasa93 are used as two "CC subsets". So, Nasa93 should be identified as very useful for predicting Nasa60 projects by any approach or analysis performed with that purpose.

Cocomo81 is used as a single CC subset for DCL as shown in table 1, instead of being divided into three. Similar to Nasa60Coc81, the original order of the Nasa60 projects is preserved in order to simulate the WC data stream. Note that Nasa60Coc81Nasa93 is thus composed of two CC subsets (Cocomo81 and Nasa93) and one WC data stream (Nasa60).

It is worth noting that there are mainly two types of analyses in this paper: (1) analyses to evaluate and understand the behaviour of DCL and its ability to emphasize the right models, and (2) analysis to check whether CC data can improve predictive performance over WC models. For the former, the existence of an overlap between the CC and WC data simulates the case where a given CC model is very helpful for improving WC predictions. This is very useful to test whether DCL is successful in finding out that the corresponding CC model is helpful. For the latter, the overlap means that we cannot use Nasa93 as a CC subset to draw the conclusion that CC data are useful for improving WC predictions.

6 DCL's Ability to Emphasize the Right Models

DCL was designed to be able to identify which past models best represent the current projects being estimated by a given company in terms of the effort required to develop software projects. Therefore, it is essential to analyse whether DCL is successful in doing so. This section evaluates the ability of DCL's dynamic weighting mechanism to emphasize the models that best represent the company that we are interested in (single company). It investigates how well DCL addresses RQ1. It also provides an in depth understanding of DCL's behaviour, contributing to its external validity.

6.1 Experimental Setup

The analysis is based on the following:

- the weights given by DCL to each base learner, and
- the predictive performance of each base learner in isolation, i.e., when used for predictions as a single model rather than within DCL.

If the highest / lowest weights correspond to the base learners that perform best / worst, DCL's weighting scheme is successful.

Regression trees (RTs) were used as the data models in the experiments, as they achieved good predictive performance for SEE in comparison to several

other approaches [44]. The input and output attributes of the RTs are the input and output attributes of the datasets explained in section 5. We used the REPTree implementation from Weka [19] to implement the RTs. The parameters of the RTs were minimum total weight of one for the instances in a leaf, and minimum proportion 0.0001 of the variance on all the data that need to be present at a node in order for splitting to be performed. These parameters were shown to be appropriate in the literature [48].

DCL's parameters were the default values of $\beta_c = \beta_w = 0.5$, $T_{start} = 15$ and $Q = 0.9$. The use of default values for DCL ensures that any benefit obtained by DCL does not depend heavily on fine tuning its parameters. This is especially desirable considering that companies would frequently not have resources for fine tuning parameters, and that non-stationary environments might cause the best parameters to change with time. An analysis of the impact of different parameter choices on DCL's predictive performance is provided in section 10.

A single execution for each dataset from section 5 was performed, as both DCL and the RTs used in this study are deterministic and the datasets must have their order of examples fixed to represent the real online learning scenario of a company. This is a standard procedure when evaluating online learning approaches [49,31].

At each time step, the predictive performance was measured in terms of Mean Absolute Error (MAE) over the predictions on the next ten projects of the WC data stream. MAE is defined as:

$$MAE = \sum_{i=1}^{n} \frac{|y_i - \hat{y}_i|}{n},$$

where n is the number of cases considered (which is ten in this work), 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 unbiased towards under or overestimates, different from other measures such as measures based on the Magnitude of the Relative Error (MRE) [37,56]. Lower MAE indicates higher/better predictive performance.

6.2 Analysis

If a base learner performs better than another one at a given time step, this means that it represents the current relationship between input and output attributes of the single company better than this other model. Therefore, DCL's weighting scheme can be considered successful in learning this if it assigns higher weights to base learners that perform better. In particular, if a certain CC model obtains better predictive performance than the WC model at a given time step, it is considered more beneficial than the WC model at this time step. If the CC model's predictive performance is worse at a given time step, it is considered detrimental at this time step. Therefore, we provide an

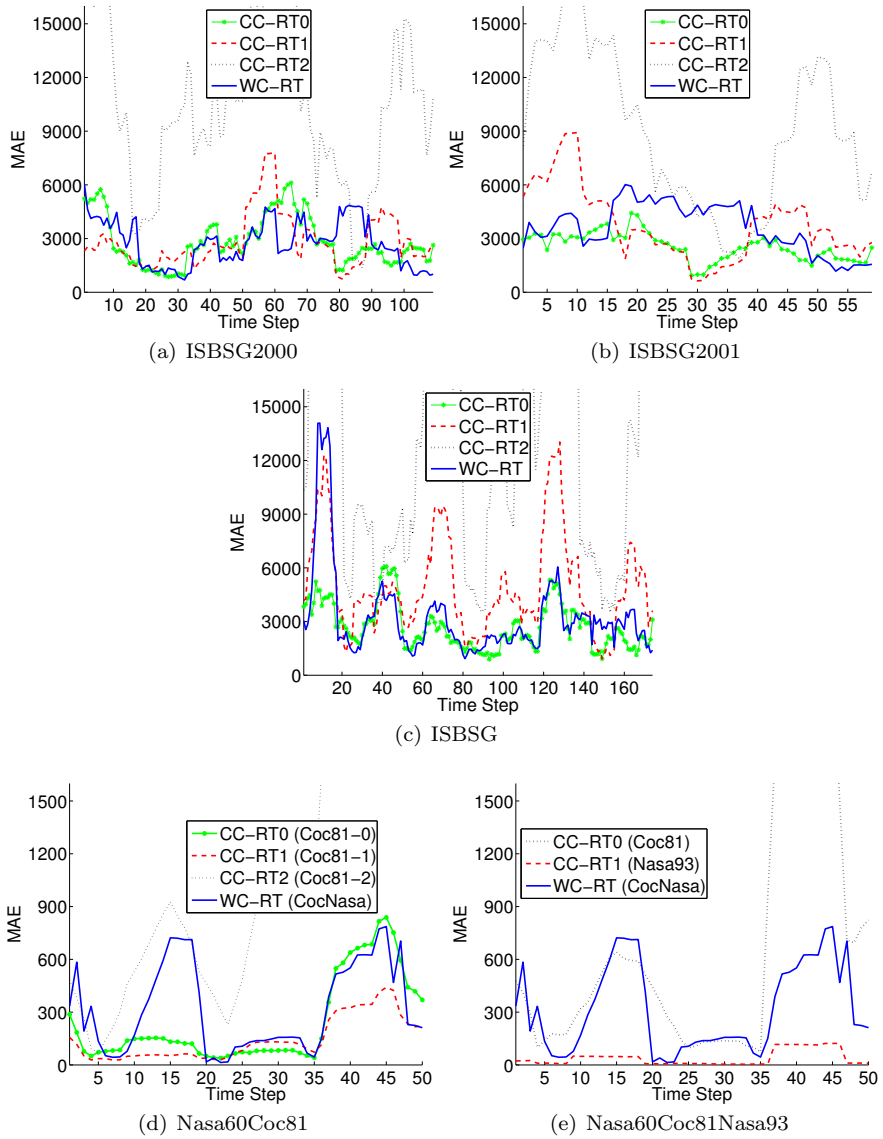
analysis of the predictive performance of each RT used within DCL in combination with the weight assigned by DCL to it.

Figure 2 shows the MAE of each isolated RT at each time step. We can see that the MAEs of different models (CC-RTs and WC-RT) are considerably different from each other and that they can become more or less beneficial/detrimental for the single company whose projects are being estimated, depending on the moment in time. For ISBSG2000, ISBSG2001, ISBSG and Nasa60Coc81 (the datasets that use real CC data), it is interesting to observe that the CC-RTs sometimes perform better than the WC-RT. This demonstrates that CC models have the potential to improve predictive performance over WC models [43].

Figure 3 shows the weights attributed by DCL to each model throughout time. The weights vary much more for ISBSG2000, ISBSG2001 and ISBSG than for Nasa60Coc81 and Nasa60Coc81Nasa93. This reflects the fact that the relative predictive performance of the best performing base learners is less stable for ISBSG2000, ISBSG2001 and ISBSG than for Nasa60Coc81 and Nasa60Coc81Nasa93 (figure 2). For instance, the predictive performances of CC-RT0 and WC-RT are competing during several moments for ISBSG (figure 2(c)), especially after time step 50. These two models usually present the best predictive performances after this time step (figure 2(c)). This is reflected by DCL, whose weights attributed to CC-RT0 and WC-RT are also dominating and competing against each other during this period (figure 3(c)). On time steps 20-45, CC-RT1 also achieves competitive predictive performance (figure 2(c)), presenting similar and competing weight to CC-RT0 and WC-RT (figure 3(c)). Before time step 20, the best predictive performance is achieved by CC-RT0, and this is also successfully reflected by its higher weight during the first time steps. Nevertheless, there are some moments in time where a certain learner is best at predicting the next ten projects, but this is still not reflected by DCL's weights.

From figure 2(d), we can see that CC-RT1 was the model with the best overall predictive performance across time steps for Nasa60Coc81. This was reflected by its large weight shown in figure 3(d). Therefore, DCL was successful in identifying the model that most contributed to a better overall predictive performance across time steps. However, we can also see that CC-RT0 was better than CC-RT1 between time steps 25–35, and this behaviour was not reflected by DCL's weights. A possible reason for that is that CC-RT0's weight became too low due to its prolonged time behaving worse than CC-RT1. So, we have performed experiments restricting the weights of all models to a minimum of 0.01 to check whether the weight can reflect better the current projects being estimated. This resulted in an increase of CC-RT0's weights, but this increase was delayed and only started to become more apparent after time step 30. So, restricting the weights to 0.01 did not provide significant improvements in predictive performance.

In order to find the reason why DCL is sometimes not able to emphasize the right models sufficiently, we examined the absolute error obtained by base learners on the project at the current time step. For example, in order



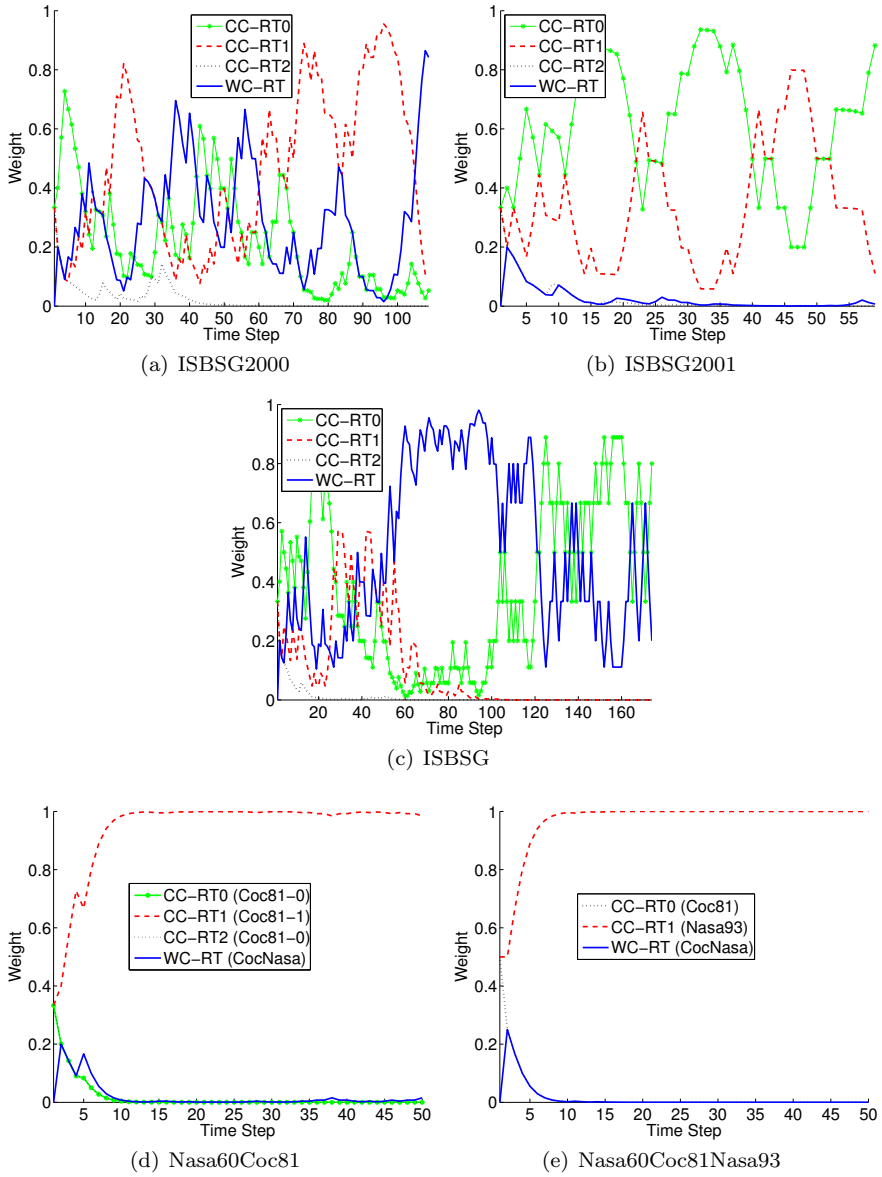
(a) ISBSG2000 (b) ISBSG2001

(c) ISBSG

(d) Nasa60Coc81 (e) Nasa60Coc81Nasa93

Fig. 2 Predictive performance of CC-RTs (not trained with WC data) and WC-RTs for each dataset in terms of MAE. Some RTs have very high MAE, but the limit of the y-axis was not increased to avoid hindering visualization of the better performing RTs. From [43].

to understand why DCL was not able to emphasize CC-RT0 sufficiently for Nasa60Coc81 between time steps 25–35, we examined the absolute error obtained by each model on the project at the current time step between time steps 25–45 (table 2). This reveals that, even though CC-RT0 is a better model to predict the effort of the *next ten projects* (figure 2(d)), WC-RT is better at



(a) ISBSG2000 (b) ISBSG2001

(c) ISBSG

(d) Nasa60Coc81 (e) Nasa60Coc81Nasa93

Fig. 3 DCL's dynamic weights.

making predictions for the *current* project during 25–27, becoming frequently the winner during these time steps. As a result, WC-RT's weight, rather than CC-RT0's, increases on these times steps (this increase is only visually noticeable in plots when restricting DCL's weights to 0.01, which were omitted due to space constraints). This analysis shows that, when future projects be-

Table 2 Absolute error of each base learner for the project at the current time step for Nasa60Coc81. Cells in yellow (light grey) represent the winner model.

Time Step	C-RT-0	CC-RT1	CC-RT2	WC-RT
25	105.00	105.57	766.00	47.00
26	77.93	62.60	119.40	2.40
27	77.93	62.60	119.40	1.20
28	21.20	6.89	56.80	10.80
29	15.20	0.89	62.80	23.80
30	1.60	15.91	79.60	9.60
31	0.80	13.51	77.20	2.40
32	26.00	11.69	52.00	7.80
33	177.80	151.80	70.20	52.80
34	149.00	264.50	2131.00	857.00
35	185.00	228.50	2095.00	120.00
36	40.00	373.50	2240.00	145.00
37	185.00	228.50	2095.00	36.00
38	38.00	7.00	501.20	12.00
39	10.33	21.33	956.50	110.00
40	10.00	5.00	956.50	58.40

come suddenly too different from the current ones, DCL may not be able to emphasize the best models for predicting those future projects fast enough. This is not unexpected, because a learner cannot learn what has not been taught. Such a situation might be better handled by techniques for detecting concept drifts [49]. Similarly, on time steps 30-37, both CC-RT0 and WC-RT win frequently, causing their weights to increase competitively during this period. This impedes CC-RT0 of achieving a higher overall weight. The reason for WC-RT's higher MAE on time steps 25–35 (figure 2(d)) is its very high error on a single project (time step 34), and not the number of times that it loses.

From figure 2(e), we can see that CC-RT1 was the best model throughout the whole learning for Nasa60Coc81Nasa93. This is expected, as CC-RT1 was trained on a dataset containing an overlap with the WC data stream. As shown by figure 3(e), DCL gave the highest weight to this model throughout the learning. Therefore, DCL was successful in identifying CC-RT1 as the best model to be used.

In short, the dynamic weighting mechanism of DCL is generally successful in identifying the best base learners to be emphasized (RQ1). However, abrupt changes can sometimes cause a given model to suddenly present much better / worse predictive performance than before, without a transition period for its weights to gradually start reflecting the new situation. In these cases, weights have a short delay in reflecting the new situation, because they can only start reflecting it once it becomes active. In the future, techniques for detecting concept drifts [49] could be used here.

7 DCL's Predictive Performance

DCL was designed to emphasize the models that reflect well the current relationship between input and output attributes for a given company. Section 6 demonstrated that DCL is successful in doing so. However, no analysis has been done so far to check whether emphasizing the right models can really lead to improvements in SEE. This section presents such an analysis, contributing to answering RQ2. Given that DCL relies not only on the weighting mechanism, but also on a backup filtering mechanism, this section also investigates how helpful each of these mechanisms is in improving predictive performance and whether they are both really essential to DCL. Section 7.1 explains the experimental setup, and sections 7.2 and 7.3 present the analysis.

7.1 Experimental Setup

The analysis presented in this section is based on a comparison between DCL and the following approaches:

- DCL using only filtering (DCL-F),
- DCL using only dynamic weighting (DCL-W) and
- DCL using no dynamic weighting and no filtering (DCL-N).

DCL-F and DCL-N represent approaches that give the same emphasis to all SEE models. They use a fixed weight of $1/(m+1)$ for each model and will be used to validate DCL's success in improving SEE in comparison to corresponding approaches that do not attempt to emphasise different models. DCL-W and DCL-F will be compared to DCL in order to analyse the contribution of dynamic weighting and filtering to DCL's predictive performance. The parameters of all approaches were the same as in section 6.1.

Besides evaluating predictive performance in terms of MAE, this section also analyses the standardized accuracy (SA) and effect size Δ [56] in order to provide interpretable results in terms of the magnitude of the predictive performance. SA is defined as $SA_L = (1 - MAE_L/\overline{MAE}_R) \times 100$, where L is the approach being evaluated, MAE_L is the MAE of this approach, and $\overline{MAE}_R$ is the MAE of a large number, typically 1000, runs of random guesses (rguess). Rguess estimates the effort of a WC project p_t on a time step t as the effort of a WC project $p_i, 1 \leq i < t$ sampled uniformly at random. We used 1000 runs of rguess, following previous work [56]. SA_L is viewed as the ratio of how much better L is than rguess. The effect size Δ_C of an approach against the control approach C in terms of MAE is defined as $\Delta_C = (MAE_C - MAE_L)/S_C$, where S_C is the sample standard deviation of the control approach. As suggested by Shepperd and McDonell [56], we interpret the absolute value of the effect size, which is standardised (i.e., scale-independent), in terms of the categories proposed by Cohen [15]: small (≈ 0.2), medium (≈ 0.5) and large (≈ 0.8). So, the effect size can be used to explain how large the difference in MAE between

an approach and a control approach is, and thus gives insight into how large the impact of this difference is likely to be in practice.

SEE is typically a very difficult task and approaches might perform worse than rguess [56]. An approach that performs similar or worse than rguess is useless from the practical point of view. We compared DCL against rguess in our previous work [43]. Our experiments showed that DCL always performed statistically significantly better than random guess with very high effect size.

7.2 DCL's Ability to Improve Overall MAE

In order to investigate the success of DCL in improving SEE predictive performance, we compare it against DCL-F and DCL-N, which represent approaches that give the same emphasis to all SEE models. In order to find out which mechanism (dynamic weighting or filtering) contributes more to DCL's predictive performance, we compare DCL against DCL-W and DCL-F.

Table 3 shows the overall predictive performance achieved by DCL, DCL-W, DCL-F and DCL-N for each dataset. As we can see, DCL-W provided the best results for most datasets, whereas DCL using both dynamic weighting and filtering produced the best results for ISBSG2000. In most cases, the worst results were obtained when neither dynamic weighting nor filtering was used (DCL-N).

In order to check whether these differences in predictive performance are statistically significant, we performed a Friedman test for comparing the overall MAE across multiple datasets, as recommended by Demšar [17]. The test detected statistically significant difference at the level of significance of 0.05 ($F_F = 27.25 > F(3, 12) = 3.49$, p-value < 0.0001). The average ranking of the approaches is shown in table 4. The highest ranked approach was DCL-W; the second highest ranked was DCL; and the worst ranked approach was DCL-N.

The average rankings give us insight into what approaches performed better/worse. For instance, even before performing post-hoc tests we can see that although DCL-W's average ranking was 1.2 and DCL's was 1.8, these two values were quite similar to each other if we consider the standard deviation. In the same way, DCL-F's and DCL-N's rankings were similar to each other, and worse than DCL-W's. Post-hoc tests with Holm-Bonferroni corrections were performed to confirm which approaches are statistically significantly different from the highest ranked approach DCL-W. The z and p-values of the post-hoc tests are shown in table 4. They confirm that DCL using both dynamic weighting and filtering performed similarly to DCL-W, whereas DCL-F and DCL-N performed worse.

These results show that dynamic weighting was essential to DCL's predictive performance, as the best results for each particular dataset were always achieved when dynamic weighting was used (DCL-W or DCL). This means that DCL's mechanism to emphasize different models is successful in improving predictive performance in comparison to approaches that give the same weight to all models. The fact that there is no statistically significant difference between DCL-W and DCL across datasets means that filtering used in combination with weighting did not provide additional benefits in all cases.

Table 3 Overall predictive performance averaged across time steps. Cells in yellow (light grey) represent the best values.

MAE +- Std Dev				
Dataset	DCL	DCL-W	DCL-F	DCL-N
ISBSG2000	2352.59 +- 925.84	2554.36 +- 1073.43	2641.08 +- 953.42	3521.61 +- 1632.34
ISBSG2001	2873.11 +- 1235.93	2795.39 +- 1254.17	3417.44 +- 1648.90	3573.21 +- 1589.43
ISBSG	2805.56 +- 1468.20	2741.18 +- 1396.35	3041.07 +- 2140.21	5108.66 +- 2955.82
Nasa60Coc81	205.83 +- 214.70	142.31 +- 122.64	303.28 +- 203.33	298.50 +- 196.10
Nasa60Coc81Nasa93	109.82 +- 154.72	42.25 +- 45.42	241.75 +- 185.77	251.35 +- 192.76
SA				
Dataset	DCL	DCL-W	DCL-F	DCL-N
ISBSG2000	46.21	41.60	39.61	19.48
ISBSG2001	30.14	32.03	16.90	13.11
ISBSG	53.69	54.75	49.80	15.67
Nasa60Coc81	56.92	70.22	36.52	37.52
Nasa60Coc81Nasa93	77.01	91.16	49.40	47.39
Δ_{rguess}				
Dataset	DCL	DCL-W	DCL-F	DCL-N
ISBSG2000	3.21	2.89	2.75	1.35
ISBSG2001	2.43	2.58	1.36	1.06
ISBSG	1.55	1.58	1.44	0.45
Nasa60Coc81	0.97	1.20	0.62	0.64
Nasa60Coc81Nasa93	1.32	1.56	0.84	0.81

Table 4 Ranking average and standard deviation of approaches across datasets based on overall MAE; and z and p-values of the post-hoc tests for comparison of each approach against DCL-W. 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. For each dataset, smaller ranking represents better overall MAE.

Approach	Rank Avg	Rank Std	z	P-value
DCL-W	$1.2 \approx 1$	0.45	–	–
DCL	$1.8 \approx 2$	0.45	0.7348	0.4624
DCL-F	$3.2 \approx 3$	0.45	2.4495	0.0143
DCL-N	$3.8 \approx 4$	0.45	3.1843	0.0015

7.3 Analysis of MAE at Each Time Step

Statistics such as overall average predictive performance reported in section 8.2 are good for providing a general idea of the predictive performance of approaches. However, drawing conclusions based solely on such statistics is not ideal, as they may hide other characteristics of the behaviour of the prediction models throughout time that can be important when choosing one model over the other. For instance, these statistics do not show whether a certain approach is better at some time steps, but worse at others.

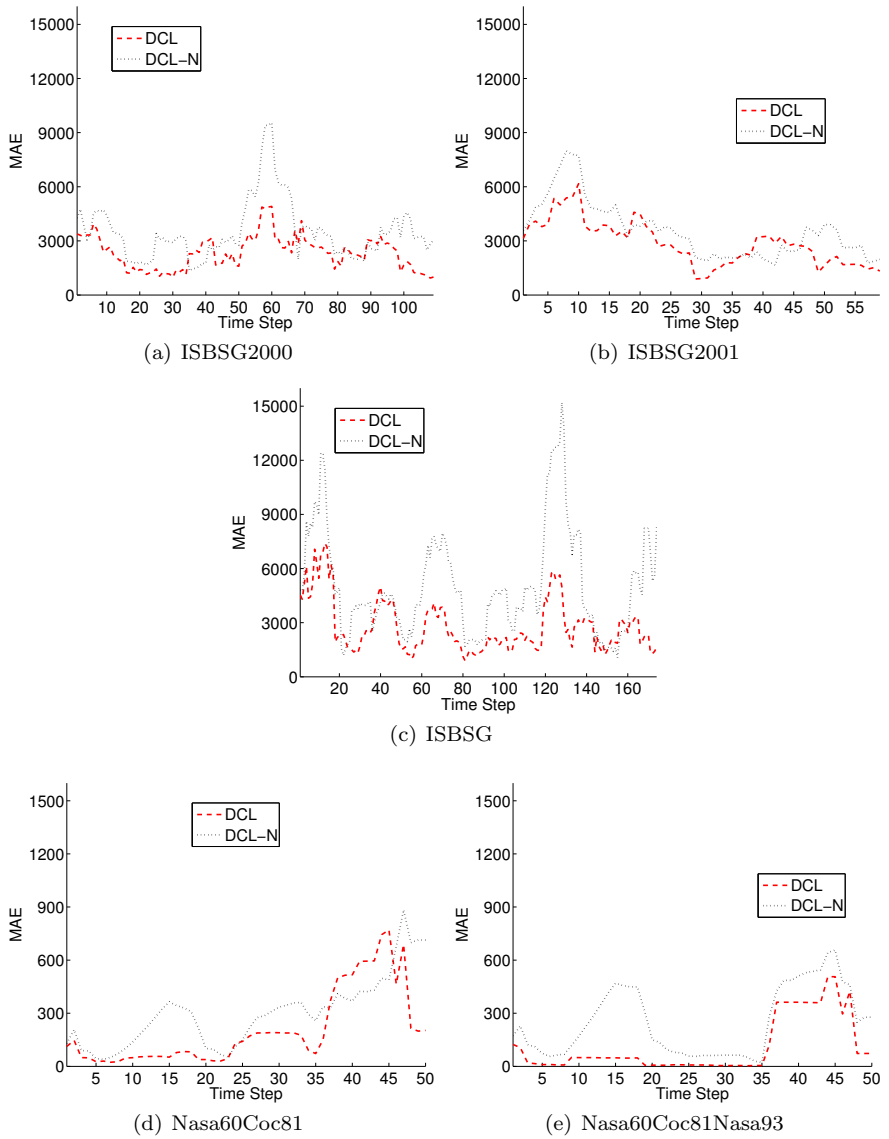
Figure 4 shows DCL's MAE throughout time against DCL-N's, which is an approach that does not use DCL's filtering and adaptive weighting mechanisms. We can see that DCL performed better than DCL-N most of the time, reflecting its better overall MAE presented in section 7.2. During a few periods of time, however, DCL-N outperformed DCL. These were around time step 40 for ISBSG2000, 40–45 for ISBSG2001 and 37–45 for Nasa60Coc81. Improving DCL's weighting mechanism may help to avoid such periods of lower predictive performance as is proposed as future work.

8 Using CC Models to Improve SEE

As explained in section 1, given the problems caused by small WC datasets, both industry and academia have been investing in CC data [54, 20], making approaches able to use such CC data to improve SEE desirable. However, existing approaches in the literature struggle to achieve such improvements in predictive performance (see section 2). This section investigates how DCL's predictive performance compares to WC SEEs. It also compares DCL's results against existing CC SEE approaches, to check whether it is worth adopting DCL in comparison to existing approaches. Together with section 7, this section answers RQ2. Section 8.1 explains the experimental setup and sections 8.2 and 8.3 present the analysis.

8.1 Experimental Setup

DCL and the following approaches were compared:



(a) ISBSG2000 (b) ISBSG2001

(c) ISBSG

(d) Nasa60Coc81 (e) Nasa60Coc81Nasa93

Fig. 4 Predictive performance of DCL and DCL-N for each dataset in terms of MAE. At each time step, a new WC project is used for training, then the approaches are used to predict the next ten projects, and the MAE is calculated based on these ten predictions.

– RT trained on the WC data stream (referred to simply as RT). As explained in section 3, a new WC project is completed and received at each time step. In order to create RTs, at each time step, the current RT was discarded and a new RT was trained on all WC projects completed so far (including the one completed at the current time step). This RT was then used to predict

future projects of the WC data stream. As explained in section 6.1, RTs have shown to produce good predictive performance for SEE in comparison to several other automated approaches [44], being a good baseline approach for this study.
- RT trained both on the CC and WC data streams (CC-RT). At each time step, the current CC-RT was discarded and a new CC-RT was trained on all CC data and WC projects completed so far (including the one completed at the current time step). This CC-RT was then used to predict future projects of the WC data stream. This approach represents existing CC techniques that treat CC and WC data as a single dataset to be used for building SEE models, e.g., [33].
- Relevancy filtering [61] trained on the WC data stream (Relevancy Filtering). Relevancy filtering is the state-of-the-art in CC SEE. As it can be applied to both WC and CC data, we first compare DCL against a WC-only Relevancy Filtering using RT as the base learner. Similar to the previous approaches, at each time step, the current Relevancy Filtering model was discarded and a new one trained on all WC projects completed so far.
- Relevancy filtering [61] trained on the WC and CC data (CC-Relevancy Filtering). This model is created similarly to WC Relevancy Filtering, but using not only all WC data received so far, but also all CC data as the training data at each time step.
- DWM trained on the WC data stream (referred to as DWM). DWM is a popular ML approach for dealing with non-stationary environments [31]. It is included in the analysis to check whether DCL would be able to improve predictive performance over an existing dynamic adaptive approach and was first investigated in the SEE context in [43]. Different from DDD [49], DWM keeps base learners likely to represent several different concepts. This is more likely to be beneficial for SEE, where each concept is available for a short period of time and may reoccur in the future.
- DWM first trained using the CC and then the WC data stream (CC-DWM). CC-DWM was used to check whether DCL would produce competitive results in comparison to a CC approach prepared to deal with non-stationary environments.

The weight update rule used in 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 learner is added to the ensemble if its estimation on the current training project has absolute error higher than τ in a time step multiple of p, where p is a parameter of DWM/CC-DWM. Existing base learners with weight $< \theta$ are deleted also in time steps multiple of p, following the original DWM algorithm [31].

DCL's, CC-RT's and RT's parameters were the default parameters shown in section 6.1. This ensures that the analysis of the behaviour of these approaches does not depend on fine tuning parameters. Relevancy Filtering and CC-Relevancy Filtering used the default parameter value of $k = 10$ [61]. DWM

and CC-DWM used the default parameter values of $\beta = 0.5$, $p = 1$, $\theta = 0.01$ and $\tau = 0.25y$ [31,43] [4].

A single execution for each dataset from section 5 was performed for RT, CC-RT, DWM, CC-DWM and DCL as they are deterministic when using the deterministic RTs in this study. CC-Relevancy Filter and Relevancy Filter are non-deterministic. So, 30 runs were performed for these approaches. The average of the thirty runs at each time step was used in the analysis.

This analysis depends not only on DCL's ability to emphasize the right models, but also on how much CC data can help improving predictive performance in comparison to WC models. Given that Nasa60Coc81Nasa93 contains a CC subset which overlaps with the WC data, it cannot be used in this analysis. Therefore, this analysis is based only on ISBSG2000, ISBSG2001, ISBSG and Nasa60Coc81.

8.2 Analysis of Overall MAE Across Time Steps

Table 5 presents the overall predictive performance across time steps in terms of MAE, SA and Δ_{rguess}. RT, DWM, Relevancy Filtering and CC-Relevancy Filtering have effect size Δ_{rguess} varying from medium to large, whereas DCL and CC-DWM always have a very high effect size, showing a much better predictive performance. Even though DCL's, CC-DWM's and CC-Relevancy Filtering's SAs were considerably larger given the difficulty of the SEE task, RT, DWM and Relevancy Filtering presented low SA for ISBSG2001, indicating a clear need for improvement.

In order to compare multiple models over multiple datasets, we used Friedman statistical tests, as recommended by Demšar [17]. 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 ($F_F = 6.46 > F(6, 18) = 2.66$, p-value = 0.0009). The ranking of approaches obtained from the test is shown in table 6. DCL was ranked first (lowest/best MAE) for all datasets and was the only approach ranked higher than RT. Therefore, we computed Wilcoxon Signed-Rank tests with Holm-Bonferroni between DCL and RT for each dataset with overall level of significance of 0.05. This is a stronger test than the usual post-hoc tests that would normally be needed if we were interested in comparing every approach against the baseline RT. Statistical comparisons between other approaches and RT are not needed in this case because we know from the rankings that none of them would outperform RT. The p-values for comparing DCL against RT are 0.0015 for ISBSG2000, 0.0139 for ISBSG2001, < 0.0001 for ISBSG and 0.0074 for Nasa60Coc81, confirming that DCL outperforms RT on all these datasets.

[4] DCL was compared against DWM and CC-DWM in our preliminary work [43], but those experiments were performed fine tuning DWM's and CC-DWM's parameters

Table 5 Overall predictive performance averaged across time steps. Cells in yellow (light grey) represent the highest ranked values.

MAE +- Std Dev							
Dataset	DCL	RT	CC-RT	Relevancy Filtering	CC-Relevancy Filtering	DWM	CC-DWM
ISBSG2000	2352.59 +- 925.84	2753.37 +- 1257.46	3271.01 +- 1887.27	2665.66 +- 1045.36	3092.01 +- 1345.41	3022.11 +- 1185.23	2906.75 +- 1076.84
ISBSG2001	2873.11 +- 1235.93	3621.96 +- 1367.96	3417.17 +- 2070.06	3644.82 +- 1364.35	3630.93 +- 1403.04	3400.3 +- 1001.87	3336.35 +- 960.28
ISBSG	2805.56 +- 1468.20	3253.93 +- 2476.05	5244.59 +- 4047.2	3383.12 +- 2387.59	4484.34 +- 3060.12	3805.19 +- 2449.27	3812.13 +- 2457.67
Nasa60Coc81	205.83 +- 214.70	319.46 +- 250.23	578.98 +- 665.27	325.4 +- 262.94	851.61 +- 1705.34	319.54 +- 303.82	384.04 +- 331.00
SA							
Dataset	DCL	RT	CC RT	RelFilter	CC RelFilter	DWM	CC DWM
ISBSG2000	46.21	37.05	25.21	39.05	29.30	30.90	33.54
ISBSG2001	30.14	11.93	16.91	11.37	11.71	17.32	18.87
ISBSG	53.69	46.29	13.43	44.16	25.98	37.19	37.08
Nasa60Coc81	56.92	33.14	-21.18	31.89	-78.24	33.12	19.62
Δ_{rguess}							
Dataset	DCL	RT	CC RT	RelFilter	CC RelFilter	DWM	CC DWM
ISBSG2000	3.21	2.58	1.75	2.71	2.04	2.15	2.33
ISBSG2001	2.43	0.96	1.36	0.92	0.94	1.39	1.52
ISBSG	1.55	1.34	0.39	1.28	0.75	1.08	1.07
Nasa60Coc81	0.97	0.57	-0.36	0.54	-1.34	0.57	0.34

Table 6 Ranking average and standard deviation of approaches across datasets based on the overall MAE. For each dataset, smaller ranking represents better overall MAE.

Approach	Rank Avg	Rank Std
DCL	1.00	0.00
RT	3.00	1.41
DWM	$3.75 \approx 4$	0.96
Relevancy Filter	4.00	2.16
CC-DWM	4.00	1.41
CC-RT	6.00	1.41
CC-Relevancy Filter	$6.25 \approx 6$	0.50

Our experiments are based on predicting the next ten projects in the WC data stream. However, our approach is also applicable to other numbers of future projects. As a sanity check, we have performed additional experiments using DCL and the baseline approach RT for predicting the next three, five and fifteen projects using Nasa60Coc81, ISBSG, ISBSG2000 and ISBSG2001. If we count the number of wins (without checking for statistical significance), DCL wins in all cases. A Wilcoxon Signed-Rank test to compare DCL-RT and RT across datasets and numbers of future predictions confirms that DCL was better ranked than RT (p-value of 0.00044).

Overall, we can conclude that DCL is successful in using CC data to improve SEE in comparison with other existing SEE approaches, including WC approaches. This result is of practical importance. It means that companies can use data from other companies in order to reduce the issues caused by the small size of their WC SEE datasets, saving the cost of collecting a large number of WC data.

8.3 Analysis of MAE at Each Time Step

In this section, we analyse the predictive performance of DCL against the baseline approach RT in order to further understand the benefit of using DCL as a CC approach. Figure 5 shows the MAE at each time step for DCL and 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.

We can see from figures 2 and 5 that DCL managed to use the potential benefit from CC data in several cases. However, even though DCL rarely obtained worse predictive performance than RT throughout time, it still has room for further improvement. Its MAE was a bit worse during the first 15 time steps for ISBSG2001. The full potential benefit from CC-RT1 and CC-RT2 was not used by DCL in the last 15 time steps for Nasa60Coc81 either. Improvements on DCL's weighting scheme may help it to achieve better predictive performance, as it sometimes does not reflect the best models (section 6). Filtering may sometimes also hinder predictive performance, as shown in

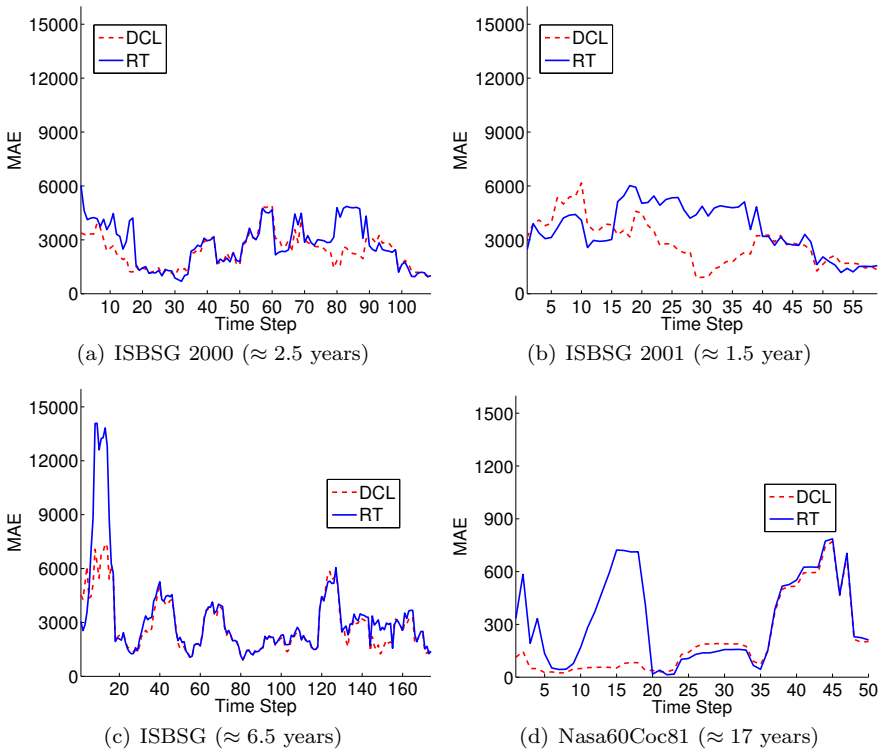


Fig. 5 Predictive performance of RT and DCL for each dataset in terms of MAE. The number of years represents the period covered by the time steps considering the implementation date of the single company projects.At each time step, a new WC project is used for training, then the approaches are used to predict the next ten projects, and the MAE is calculated based on these ten predictions.

section 7. Therefore, future research could also look into improving DCL's filtering.

We also computed the effect size of DCL against RT's for a sliding window representing different periods of time using the function shown in algorithm 2. The window size is a critical parameter of this function. Too small windows could be more likely to find short consecutive periods in which one approach is continuously better than the other, creating too large effect sizes. Too large windows would provide the same information as the overall effect size. In order to determine the window size, we ran algorithm 2 with window sizes from 10 to 2/3 of the size of the dataset and computed the average of the effect sizes in $\overrightarrow{\Delta}_{RT}$ for each window size. The average of all effect sizes in $\overrightarrow{\Delta}_{RT}$ should be close to the overall effect size. So, the window size whose average effect size achieved the smallest absolute difference with respect to the overall effect size was chosen. This resulted in window sizes of 33 for ISBSG2000, 39 for ISBSG2001, 43 for ISBSG and 32 for Nasa60Coc81. The sliding window effect

sizes are shown in figure 6. As we can see, all datasets had at least some moments in which the effect size was large and in favour of DCL (above the top horizontal dotted black line). This further demonstrates that it is worth using DCL as a CC learning approach.

Algorithm 2 Sliding Effect Size

Parameters:
$\overrightarrow{MAE}_{DCL}$: vector with DCL's MAE at each time step;
$\overrightarrow{MAE}_{RT}$: vector with RT's MAE at each time step;
T: number of time steps;
W: window size.

Create empty vector of effect sizes $\overrightarrow{\Delta}_{RT}$.
for $i = 1$ to T **do**
  if $i + W - 1 > T$ **then**
    Return Eff.
  $AvgDcl \leftarrow \frac{\sum_{j=i}^{j=i+W-1} \overrightarrow{MAE}_{DCL}[j]}{W}$
  $AvgRt \leftarrow \frac{\sum_{j=i}^{j=i+W-1} \overrightarrow{MAE}_{RT}[j]}{W}$
  $StdRt \leftarrow \sqrt{\frac{1}{W-1} \sum_{j=i}^{j=i+W-1} (\overrightarrow{MAE}_{RT}[j] - AvgRt)^2}$
  Add $\frac{AvgRt - AvgDcl}{StdRt}$ to the end of $\overrightarrow{\Delta}_{RT}$.
Return $\overrightarrow{\Delta}_{RT}$.

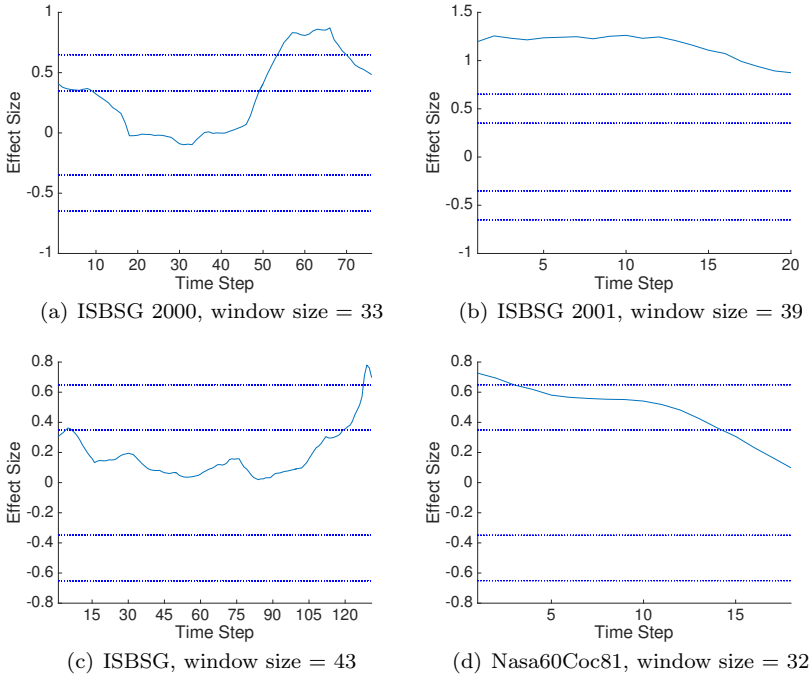
9 DCL and Types of Base Learners

This section analyses the overall behaviour of DCL when using different types of base learner than RTs. It aims at (1) verifying whether it can still provide benefits over its corresponding WC model when using other types of base learner, and at (2) revealing what base learners do better in combination with DCL. This analysis supports the external validity of this work, as it shows whether DCL still performs as expected even when using different base learners. It also gives insights to practitioners in terms of what base learners to use with DCL for achieving better SEEs.

9.1 Experimental Setup

Besides using RTs as in the previous sections, this section also considers the following base learners:

- Estimation by Analogy (EBA) [57] – this is the k-nearest neighbour algorithm [9] based on normalised attributes and Euclidean distance.
- Bagging ensembles of RTs (Bag+RTs) [44].
- Radial Basis Function networks (RBFs) [9].
- Multilayer Perceptrons (MLPs) [9].



(a) ISBSG 2000, window size = 33 (b) ISBSG 2001, window size = 39

(c) ISBSG, window size = 43 (d) Nasa60Coc81, window size = 32

Fig. 6 Effect size Δ_{RT} based on sliding windows. The horizontal dotted black lines represent the borders between what is considered a positive or negative small/medium/large effect size.

These base learners were chosen because they provide a variety of different behaviours, including EBA as a competitive traditional approach to SEE; Bag+RTs and RTs as two approaches that have more recently been shown to be good for SEE in comparison to other approaches [44]; MLP as an approach that has been shown to be successful as a base learner in other types of ensembles for SEE [44,50], and RBF as an approach that has not been doing very well for SEE [44,50]. RBFs were included to check whether DCL could make their predictive performance competitive. Similar to section 8, base learners trained on the WC data streams are simply referred to by the base learner's name (e.g., RT or RBF). DCL using a certain base learner is referred to as DCL+<base learner name> in this section. For example, we refer to the DCL models being used in the previous sections as DCL+RT in this section. Similar to section 8, a WC model used on its own is simply referred to by its name (e.g., RT).

The parameters used with DCL and RT are the default values used in section 6.1. The parameter values for the other base learners were chosen following the same procedure as for RTs, i.e., they are the values that most often produced better results in [44]. These were number of neighbours $k = 1$ for EBA; number of hidden nodes 9, learning rate and momentum 0.1, number of epochs 100 for MLP; minimum standard deviation for clusters 0.01 and

number of clusters 6 for RBF; and for Bag+RT, number of base learners 50, minimum total weight of 1 for the instances in a leaf of the RT, and minimum proportion of the variance on all the data that need to be present at a node in the RT in order for splitting to be performed 0.0001. A single execution was performed for each dataset for EBA, which is a deterministic approach as our RT. For the others, 30 runs were performed on each dataset.

The first part of the analysis aims at checking whether DCL improves upon its corresponding WC learners. For that, the average overall MAE of DCL using each type of base learner on each dataset was paired with the average overall MAE of the corresponding WC learner for each dataset. This comparison is not used to check what combinations of DCL and base learner could provide best results, but whether DCL can improve the predictive performance of each type of WC base learner. Therefore, given that Nasa60Coc81Nasa93 contains a CC subset which overlaps with the WC data, it was not used for this part of the analysis. This resulted in 20 pairs (5 types of base learner and 4 datasets). The comparison between DCL and the corresponding WC models was based on a Wilcoxon Signed-Rank test across datasets, as recommended by Demsar [17].

The second part of the analysis aims at finding out what combinations of base learner do better with DCL. For that, a Friedman test [17] was performed for comparison across all datasets considering each of the 5 combinations of DCL with a base learner. Statistics such as SA, effect size Δ_{rguess} and standard deviation of the ranking in terms of overall MAE were also used in the analysis.

9.2 Analysis of DCL Vs WC Models

The Wilcoxon Signed-Rank test to compare DCL and the corresponding WC models shows that the overall MAE rankings are statistically significantly different across datasets with level of significance of 0.05 (p-value = 0.00068). DCL's sum of ranks was 196, whereas the WC models' sum of ranks was 14, where higher means better. So, DCL's ranking is generally superior than its corresponding WC models and we can conclude that DCL is generally robust to the type of base learners. The only two cases where a WC model was ranked higher than DCL were RBF for ISBSG2001 and EBA for Nasa60Coc81.

Table 7 shows the overall MAE, SA and effect size Δ_{rguess} for the best three ranked approaches in terms of MAE for each dataset. As we can see, the best three ranked approaches were always based on DCL, i.e., DCL never lost from a WC model when it was among the best ranked approaches. Moreover, the effect size Δ_{rguess} of the best three ranked approaches in terms of MAE was always large, showing that they are considerably better than rguess. Therefore, we can conclude that DCL is generally successful in improving SEE over its corresponding WC model, contributing to its external validity. As a sanity check, we have also compared the best ranked DCL against the best ranked WC model for each dataset using Wilcoxon Signed-Rank tests with Holm-

Table 7 Predictive performance of the best three ranked approaches in terms of MAE for each dataset. Effect sizes in red (dark grey) can be considered as large.

Dataset	Approach	MAE	SA	Δ_{rguess}
ISBSG2000	DCL+MLP	2262.83	48.26	3.35
	DCL+Bag+RT	2293.03	47.57	3.31
	DCL+RT	2352.59	46.21	3.21
ISBSG2001	DCL+IBK	2661.77	35.28	2.84
	DCL+MLP	2784.57	32.29	2.60
	DCL+RT	2873.11	30.14	2.43
ISBSG	DCL+Bag+RT	2586.80	57.30	1.66
	DCL+RT	2805.56	53.69	1.55
	DCL+MLP	2856.90	52.84	1.53
Nasa60Coc81	DCL+RT	205.83	56.92	0.97
	DCL+Bag+RT	244.93	48.74	0.83
	DCL+MLP	247.40	48.22	0.82

Table 8 Ranking average and standard deviation of DCL approaches across datasets based on overall MAE; and z and p-values of the post-hoc tests for comparison of each approach against DCL+RBF. The p-values in yellow (light grey) represents statistically significant difference of overall MAE using Holm-Bonferroni corrections at the overall level of significance of 0.05. For each dataset, smaller ranking represents better overall MAE.

Approach	Rank Avg	Rank Stdev	z	p-value
DCL+RT	2	1.00	2.6	0.0093
DCL+Bag+RT	$2.2 \approx 2$	1.10	2.4	0.0164
DCL+MLP	$2.6 \approx 3$	1.14	2	0.0455
DCL+EBA	$3.6 \approx 4$	1.67	1	0.3173
DCL+RBF	$4.6 \approx 5$	0.55	–	–

Bonferroni corrections. The results show that DCL was significantly better for all datasets.

9.3 Analysis of Combinations of Base Learners and DCL

The Friedman test for comparison of DCL using different base learners in terms of overall MAE across datasets detected statistically significant difference at the level of significance of 0.05 ($F_F = 3.58 > F(4, 16) = 3.01$, p-value $= 0.0288$). Table 8 shows that DCL+RT is the highest ranked approach. From the table, we can see that the average rankings of DCL+RT, DCL+Bag+RT and DCL+MLP are similar. DCL+EBA has a larger standard deviation. This reflects the fact that even though it was the best ranked on one dataset, it was the worst ranked on two others. The approach with the worst average ranking (DCL+RBF) was more consistently the worst ranked, as we can see from its lower standard deviation. So, we performed post-hoc tests with Holm-Bonferroni corrections to check what approaches are statistically significantly different from it. The tests reveal that both DCL+RT and DCL+Bag+RT perform better than DCL+RBF, whereas DCL+MLP's and DCL+EBA's rankings are not statistically significantly different from DCL+RBF's.

Table 9 ANOVA factors and interactions.

Factors	2-Factor Interactions	3- and 4-Factor Interactions
β_w	$\beta_w * \beta_c$	$\beta_w * \beta_c * Q$
β_c	$\beta_w * Q$	$\beta_w * \beta_c * T_{start}$
Q	$\beta_w * T_{start}$	$\beta_w * Q * T_{start}$
T_{start}	$\beta_c * Q$	$\beta_c * Q * T_{start}$
	$\beta_c * T_{start}$	$\beta_w * \beta_c * Q * T_{start}$
	$Q * T_{start}$	

It is worth mentioning that Bag+RTs were always higher ranked than single RTs, corroborating Minku and Yao's results in terms of comparison of these two approaches [44]. However, DCL+RTs is very competitive, being three times better and two times worse than DCL+Bag+RT. DCL+RT was always among the three best approaches, whereas DCL+Bag+RT was not for ISBSG2001. None of the cases where DCL lost from the corresponding WC model involved RTs or Bag+RTs. RBFs obtained the worst rankings in 4 out of 5 datasets, and DCL+RBF did not manage to improve the ranking enough to be among the best three. Overall, this analysis shows that it is worth using DCL with RTs as the base learners.

10 The Impact of DCL's Parameters

The previous sections used DCL's default parameters. This section analyses DCL's sensitivity to parameters, revealing whether DCL's predictive performance can be influenced by parameter tuning [58] and which parameters influence its predictive performance the most.

10.1 Experimental Setup

We have performed an Analysis of Variance (ANOVA) based on a full factorial design. The factors analysed are the main parameters involved in DCL: Q, T_{start}, β_c, and β_w. The first two are parameters related to the filtering of cross-company models, whereas the last two are used in the adjustment of weights of base learners. This leads to a total of 15 factors and interactions, as shown in table 9. The following parameter values were tested:

- $Q = 0.6, 0.7, 0.8, 0.9, 1.0$;
- $T_{start} = 0, 15, 30, 45, 60$;
- $\beta_c = 0.1, 0.3, 0.5, 0.7, 0.9$; and
- $\beta_w = 0.1, 0.3, 0.5, 0.7, 0.9$;

For each dataset among ISBSG2000, ISBSG2001, ISBSG and Nasa60Coc81, the observations were DCL's MAEs at each time step. The base learners were RTs, as in section 8. Sphericity is the main assumption made by within-subjects ANOVA [17]. We have used Mauchly tests to check whether sphericity is violated, and found that sphericity is violated in all cases, except for

T_{start} in CocNasaCoc81 and ISBSG2000. Whenever sphericity was violated, Greenhouse-Geisser corrections have been adopted. Our analysis is also based on the effect size η^2 [34] of factors and interactions with significant impact on MAE.

10.2 Sensitivity Analysis

ANOVA reveals that all factors and interactions among factors have statistically significant impact on MAE, except for β_w for ISBSG2000, and β_c and $\beta_w * Q$ for ISBSG2001. Most p-values were smaller than 0.009. As we have 15 factors and interactions, we expect all effect sizes η^2 to be small. So, we interpret them in relative rather than absolute terms. Our smallest η^2 was 0.00006, and the largest was 0.07345. We consider all η^2 larger than 0.01000 as relatively high. Table 10 shows all factors and interactions with both significant impact on MAE and η^2 larger than 0.01000, 0.02500 and 0.05000, respectively. Cohen's suggested reference values for small, medium and large η^2 were defined for one-way ANOVA, which does not reflect our design. They are more likely to be applicable to partial η^2. Our partial η^2 varied from 0.01483 to 0.28314. However, we concentrated our analysis on η^2 because, among other advantages, η^2 is more interpretable, representing the percentage of the variance accounted for each factor / interaction [34].

Table 10 Effect size η^2 of factors and interactions with statistically significant impact on MAE.

Dataset	$\eta^2 > 0.01000$	$\eta^2 > 0.02500$	$\eta^2 > 0.05000$
ISBSG2000	β_c, Q, $\beta_w * \beta_c$	None	None
ISBSG2001	None	β_w	$\beta_w * \beta_c$
ISBSG	β_c, $\beta_w * \beta_c$	None	None
CocNasaCoc81	Q, $Q * T_{start}$	β_c, T_{start}, $\beta_w * \beta_c$	β_w

Table 10 reveals that β_c or its interaction with β_w ($\beta_w * \beta_c$) had relatively high η^2 for all datasets. This means that the choice of values for parameter β_c or the way these values interact with the values chosen for β_w normally affect DCL's MAE considerably. Factors and interactions β_w, Q, T_{start}, and $Q * T_{start}$ had relatively high η^2 for some datasets, but not for the others. Therefore, even though these parameters and interactions can affect DCL's MAE, this is not always the case. Other interactions always had very small η^2, meaning that they have little effect on DCL's MAE.

It is worth observing that Q and $Q * T_{start}$ were less important for ISBSG2001 and ISBSG according to this analysis, given that their η^2 was very small for these datasets. As these parameters are related to whether or not filtering is applied, this corroborates the results presented in Table 3, which show that the difference in SA when using (DCL) or not using (DCL-W) filtering was small. It is also worth noting that the interactions between weighting

and filtering parameters (i.e., $\beta_c * Q$, $\beta_c * T_{start}$, $\beta_w * Q$ and $\beta_w * T_{start}$) always had very low η^2. Interactions between β_c and β_w always had relatively high η^2, and interactions between Q and T_{start} sometimes had relatively high η^2. This is expected, as β_w and β_c have a strong relationship with each other, and Q and T_{start} also have a strong relationship with each other.

11 Threats to Validity

In terms of internal validity [51], we have used as default parameter settings the values that have most often achieved better results in related literature [44,31, 43,61]. It is good and user-friendly to have models perform well using default parameter settings. In practice, there are still cases where further parameter tuning can bring significant predictive performance gains. There are also still cases when new learning models are needed. We have performed a sensitivity analysis to investigate which parameters affect DCL's predictive performance the most.

Construct validity was first dealt with by using MAE as a predictive performance measure. 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 predictive performance, but also the magnitude of the differences in predictive performance and effect size. Friedman, post-hoc tests and Wilcoxon Signed-Rank tests [17] were used to show the significance of the differences in MAE.

We have provided an in depth understanding of DCL to handle external validity. Among others, the analysis explains in which situations DCL is successful, and in which situations it may fail to benefit from the best performing underlying model. Predictions for an uncertain future will always incur some risk and errors. This is the case for all software effort estimation approaches, including DCL. Our analysis shows that DCL reduces this risk in comparison to other automated approaches by learning whether changes are happening and adjusting its weights accordingly. However, it is still impossible to adjust the weights to unknown changes that have never occurred in the available training data. If a practitioner expects his/her company to suffer sudden and unknown changes frequently, then automated approaches to software effort estimation may not be the best option. If changes are not very sudden, then approaches such as DCL can adjust to these changes.

Our analysis is based on five datasets. Three datasets with known WC chronological order were used. Even though the chronological order is not known for the other two, they can still be used to simulate a WC data stream and show (1) whether DCL would be able to identity which model is more relevant and (2) whether DCL can benefit from the CC models to improve SEE. Therefore, the use of these datasets contributes to the generalisation of our results. The dataset Nasa60Coc81Nasa93 was used for checking whether our analyses and approaches are successful in identifying a subset that is known

to be very useful. Obtaining additional datasets for evaluating DCL is difficult due to our need for non-proprietary datasets with information on which projects belong to a single company among the projects of a CC dataset. If more datasets become available in the future, a replicated study should be performed. Nevertheless, the datasets used in our current study can be made available through PROMISE and ISBSG. So, researchers and companies willing to use DCL could use the same CC datasets used in this study. The subsets used in our study also covered cases where they can be potentially very helpful, competitive against WC data and detrimental. So, the subsets were diverse.

CC data were considered as fixed CC datasets by DCL. We showed that, even after some periods of time when certain CC models are not useful, they can still become useful later on. Fixed CC datasets as used here can be useful for prolonged periods of time, allowing DCL to achieve similar or better overall MAE than WC models. This means that DCL as investigated here is applicable in practice. However, fixed CC datasets may not be useful indefinitely. If the weights of the CC models become very small in comparison to the WC model, this means that the CC data are not currently helping. If this happens in practice and the accuracy of DCL's predictions is deemed low, additional CC data should be acquired and DCL re-built. The fact that DCL's MAE was usually not higher than the WC model's during the beginning of the learning period suggests that resetting DCL every several years might be a good option. CC data can also be treated as a data stream and learned online, which is an area that we would like to investigate as future work.

12 Conclusions

This paper presents a dynamic adaptive automated approach to find out when CC models are beneficial and use them to improve SEE. It provides answers to the research questions as follows:

[RQ1] How can we know which model from the past best represents the current projects being estimated?

A new dynamic adaptive automated approach called DCL was proposed to answer this question. It uses weights to dynamically and automatically determine when CC models are more or less helpful than a WC model. These weights are adjusted based on DCL's predictive performance throughout time, giving more emphasis on the most recent projects in order to track possible changes in the company's environment. The dynamic weighting mechanism of DCL was successful in identifying the best base learners to be emphasized. However, abrupt changes sometimes caused a given model to suddenly present much better / worse predictive performance than before, without a transition period for its weights to gradually start reflecting the new situation. In these cases, weights had a short delay in reflecting the new situation, because they could only start reflecting it once it became active.

[RQ2] Can that information help improving SEE?

Yes. CC models whose weights are higher are emphasised by DCL in order to improve SEE. This resulted in improvements in predictive performance in comparison to corresponding WC models and existing CC approaches from the literature. DCL's weighting mechanism was important to improve SEE, even though DCL's filtering mechanism was sometimes beneficial and sometimes detrimental.

An important practical implication of our work is that DCL frees practitioners from the need for manually determining which WC or CC past models are relevant to the present. Moreover, our results show that DCL can be used to identify when CC data is useful to improve SEE in comparison to WC models, reducing the negative effect of small WC datasets on SEE. This means that our approach saves the cost of collecting WC data, enabling companies with few WC data to use SEE models. For example, these companies can use DCL with existing CC data, e.g., from ISBSG [20] or PROMISE [54] repositories.

The SEE models provided by DCL could be used for decision-support in several ways. For instance, if the SEE produced by DCL is similar to the SEE given by an expert, both the company and its clients could see that as a reassurance that the SEE given by the expert is in line with previous relevant past projects. If the SEEs given by DCL and an expert differ considerably, this could be used to trigger further analysis of the project to clarify its likely effort, as mentioned in section 1. DCL could also assist project managers with project planning (e.g., what team expertise to use), and requirement elicitators with which requirements to implement (e.g., whether it is worth lifting or including extra memory or CPU constraints, etc) [46]. This is because practitioners could get SEEs for different sets of input attribute values for a project, gaining insights into how much the resulting effort would vary. Additionally, DCL's weights can be used to identify which CC or WC SEE model best reflects the relationship between project input attributes and required effort in a company. When associated with transparent (readable) base learners such as decision trees, this can provide insights into what project attributes are more highly correlated with effort and how these attributes interact with each other. Practitioners could potentially use that information to gain insights into how to improve productivity.

Given the encouraging results achieved by DCL, we would like to perform an empirical validation of DCL with industry as future work. This will allow us to investigate not only how good DCL's SEEs are in comparison with expert-based SEEs, but also how well DCL can contribute with software project planning and management in practice. Other directions for future research include treating CC projects as a data stream in DCL; using other types of base learners with DCL; investigating other weighting mechanisms and mechanisms to accelerate DCL's recovery when there are abrupt changes; investigating other approaches to filter CC models with the aim of improving the filtering mechanism; and performing replicated studies with more datasets, if extra datasets become available.

13 Acknowledgements

This work was supported by an EPSRC grant (No. EP/J017515/1).

References

1. Amasaki, S., Lokan, C.: The effects of moving windows to software estimation: Comparative study on linear regression and estimation by analogy. In: IWSM-MENSURA, pp. 23–32. Assisi, Italy (2012)
2. Amasaki, S., Lokan, C.: The effects of gradual weighting on duration-based moving windows for software effort estimation. In: Product-Focused Software Process Improvement LNCS, vol. 8892, pp. 63–77 (2014)
3. Amasaki, S., Lokan, C.: On the effectiveness of weighted moving windows: Experiment on linear regression based software effort estimation. Journal of Software: Evolution and Process **27**, 488–507 (2015)
4. Amasaki, S., Lokan, C.: A replication of comparative study of moving windows on linear regression and estimation by analogy. In: PROMISE, pp. 6.1–6.10 (2015)
5. Amasaki, S., Takahara, Y., Yokogawa, T.: Performance evaluation of windowing approach on effort estimation by analogy. In: IWSM-MENSURA, pp. 188–195. Nara, Japan (2011)
6. Auer, M., Biffl, S.: Increasing the accuracy and reliability of analogy-based cost estimation with extensive project feature dimension weighting. In: ISESE, pp. 147–155. Redondo Beach, CA (2004)
7. Auer, M., Trendowicz, A., Graser, B., Haunschmid, E., Biffl, S.: Optimal project feature weights in analogy-based cost estimation: Improvement and limitations. IEEE Transactions on Software Engineering **32**, 83–92 (2006)
8. Bettenburg, N., Nagappan, M., Hassan, A.E.: Think locally, act globally: Improving defect and effort prediction models. In: MSR, pp. 60–69 (2012)
9. Bishop, C.: Neural Networks for Pattern Recognition. Oxford University Press, United Kingdom (2005)
10. Bishop, C.: Pattern Recognition and Machine Learning. Springer, Singapore (2006)
11. Boehm, B.: Software Engineering Economics. Prentice-Hall, Englewood Cliffs, NJ (1981)
12. Briand, L., Langley, T., Wieczorek, I.: A replicated assessment of common software cost estimation techniques. In: ICSE, pp. 377–386. Como, Italy (2000)
13. Cartwright, M., Shepperd, M., Song, Q.: Dealing with missing software project data. In: METRICS, pp. 154–165. Sydney (2003)
14. Cherkassky, V.S., Mulier, F.: Learning from Data: Concepts, Theory, and Methods. John Wiley & Sons, Inc., New York (1998)
15. Cohen, J.: A power primer. Psychological Bulletin **112**, 155–159 (1992)
16. DeMarco, T.: Controlling Software Projects: Management, Measurement, and Estimates. Prentice-Hall PTR (1986)
17. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. JMLR **7**, 1–30 (2006)
18. Gruschke, T.M., Jørgensen, M.: The role of outcome feedback in improving the uncertainty assessment of software development effort estimates. ACM TOSEM **17**(4), 20:1–20:35 (2008)
19. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The weka data mining software: An update. SIGKDD Explorations **11**(1), 10–18 (2009)
20. ISBSG: The International Software Benchmarking Standards Group. `http://www.isbsg.org` (2011)
21. Jørgensen, M., Grimstad, S.: The impact of irrelevant and misleading information on software development effort estimates: A randomized controlled field experiment. IEEE TSE **37**(5), 695–707 (2011)
22. Jørgensen, M., Shepperd, M.: A systematic review of software development cost estimation studies. IEEE TSE **33**(1), 33–53 (2007)

23. Kitchenham, B., Mendes, E., Travassos, G.: Cross versus within-company cost estimation studies: A systematic review. IEEE TSE **33**(5), 316–329 (2007)
24. Kitchenham, B., Pfleeger, S., McColl, B., Eagan, S.: An empirical study of maintenance and development estimation accuracy. Journal of Systems and Software **64**, 57–77 (2002)
25. Kitchenham, B., Taylor, N.: Software cost models. ICL Technical Journal pp. 73–102 (1984)
26. Kocaguneli, E., Gay, G., Menzies, T., Yang, Y., Keung, J.W.: When to use data from other projects for effort estimation. In: ASE, pp. 321–324. Antwerp, Belgium (2010)
27. Kocaguneli, E., Menzies, T., Keung, J.: On the value of ensemble effort estimation. TSE **38**(6), 1403–1416 (2012)
28. Kocaguneli, E., Menzies, T., Mendes, E.: Transfer learning in effort estimation. Empirical Software Engineering pp. 31 pages, doi: 10.1007/s10,664–014–9300–5 (2014 (in press))
29. Kok, P., Kitchenham, B., Kirawkowski, J.: The mermaid approach to software cost estimation. In: ESPRIT, pp. 296–314. Brussels (1990)
30. Kolter, J.Z., Maloof, M.A.: Using additive expert ensembles to cope with concept drift. In: ACM ICML, pp. 449–456. Bonn, Germany (2005)
31. Kolter, J.Z., Maloof, M.A.: Dynamic weighted majority: An ensemble method for drifting concepts. JMLR **8**, 2755–2790 (2007)
32. Kultur, Y., Turhan, B., Bener, A.: Ensemble of neural networks with associative memory (ENNA) for estimating software development costs. Knowledge-Based Systems **22**, 395–402 (2009)
33. Lefley, M., Shepperd, M.: Using genetic programming to improve software effort estimation based on general data sets. In: GECCO, vol. LNCS 2724, pp. 2477–2487. Chicago (2003)
34. Levine, T.R., Hullett, C.R.: Eta squared, partial eta squared, and misreporting of effect size in communication research. Human Communication Research **28**(4) (2002)
35. Lokan, C., Mendes, E.: Investigating the use of chronological splitting to compare software cross-company and single-company effort predictions. In: EASE, p. 10p. Bari, Italy (2008)
36. Lokan, C., Mendes, E.: Applying moving windows to software effort estimation. In: ESEM, pp. 111–122. Lake Buena Vista, Florida, USA (2009)
37. Lokan, C., Mendes, E.: Investigating the use of chronological split for software effort estimation. IET-Software **3**(5), 422–434 (2009)
38. Lokan, C., Mendes, E.: Using chronological splitting to compare cross- and within-company effort models: Further investigation. In: ACSC, pp. 35–42. Wellington, New Zealand (2009)
39. Lokan, C., Mendes, E.: Investigating the use of duration-based moving windows to improve software effort prediction. In: Proceedings of the 19th Asia-Pacific Software Engineering Conference, pp. 818–827. Hong Kong (2012)
40. Lokan, C., Mendes, E.: Investigating the use of duration-based moving windows to improve software effort prediction: A replicated study. Information and Software Technology **56**, 1063–1075 (2014)
41. MacDonell, S., Shepperd, M.: Data accumulation and software effort prediction. In: ESEM, pp. 31.1–31.5. Bolzano-Bolzen, Italy (2010)
42. Menzies, T., Butcher, A., Cok, D., Marcus, A., Layman, L., Shull, F., Turhan, B., Zimmermann, T.: Local vs. global lessons for defect prediction and effort estimation. IEEE TSE **39**(6), 822–834 (2013)
43. Minku, L., Yao, X.: Can cross-company data improve performance in software effort estimation? In: PROMISE, pp. 69–78, doi: 10.1145/2365,324.2365,334. Lund, Sweden (2012)
44. Minku, L., Yao, X.: Ensembles and locality: Insight on improving software effort estimation. Information and Software Technology (IST) **55**(8), 1512–1528 (2013)
45. Minku, L., Yao, X.: How to make best use of cross-company data in software effort estimation? In: ICSE, pp. 446–456. Hyderabad (2014)
46. Minku, L.L., Mendes, E., Turhan, B.: Data mining for software engineering and humans in the loop. Progress in Artificial Intelligence **5**(4), 307–314 (2015)

47. Minku, L.L., White, A., Yao, X.: The impact of diversity on on-line ensemble learning in the presence of concept drift. IEEE TKDE **22**(5), 730–742 (2010)
48. Minku, L.L., Yao, X.: A principled evaluation of ensembles of learning machines for software effort estimation. In: PROMISE, pp. 10p, doi: 10.1145/2020,390.2020,399. Banff, Canada (2011)
49. Minku, L.L., Yao, X.: DDD: A new ensemble approach for dealing with concept drift. IEEE TKDE **24**(4), 619–633 (2012)
50. Minku, L.L., Yao, X.: Software effort estimation as a multi-objectice learning problem. ACM TOSEM **22**(4), Article No. 35, 32p. (2013)
51. Mitchell, M.L., Jolley, J.M.: Research Design Explained, 7th edn. Cengage Learning, USA (2010)
52. Muthukrishnan, S.: Data Streams: algorithms and applications. Now Publishers Inc., Hanover, MA (2005)
53. Premraj, R., Shepperd, M., Kitchenham, B., Forselius, P.: An empirical analysis of software productivity over time. In: METRICS, pp. 37.1–37.10. Como, Italy (2005)
54. Sayyad Shirabad, J., Menzies, T.: The PROMISE Repository of Software Engineering Databases. School of Information Technology and Engineering, University of Ottawa, Canada (2005, `http://promise.site.uottawa.ca/SERepository`,`http://openscience.us/repo/`)
55. Sentas, P., Angelis, L., Stamelos, I., Bleris, G.: Software productivity and effort prediction with ordinal regression. Information and Software Technology **47**, 17–29 (2005)
56. Shepperd, M., McDonell, S.: Evaluating prediction systems in software project estimation. IST **54**(8), 820–827 (2012)
57. Shepperd, M., Schofield, C.: Estimating software project effort using analogies. IEEE TSE **23**(12), 736–743 (1997)
58. Song, L., Minku, L., Yao, X.: The impact of parameter tuning on software effort estimation using learning machines. In: PROMISE, pp. Article No. 9, 10p., doi: 10.1145/2499,393.2499,394. Baltimore, USA (2013)
59. Stutzke, R.D.: Software Management, chap. Software Project Estimation: an overview. John Wiley & Sons Inc, Hoboken, New Jersey (2006)
60. Tibshirani, R.: Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society: Series B **58**, 267–288 (1996)
61. Turhan, B., Mendes., E.: A comparison of cross- versus single- company effort prediction models for web projects. In: Euromicro Conference on Software Engineering and Advanced Applications, pp. 285–292. Verona, Italy (2014)
62. Wen, J., Li, S., Lin, Z., Hu, Y., Huang, C.: Systematic literature review of machine learning based software development effort estimation models. IST **54**, 41–59 (2012)
63. Wieczorek, I., Ruhe, M.: How valuable is company-specific data compared to multi-company data for software cost estimation? In: IEEE International Software Metrics Symposium (METRICS), pp. 237–246. Ottawa (2002)