Multi-stream online transfer learning for software effort estimation

Minku, Leandro

DOI:
10.1145/3475960.3475988

Document Version
Peer reviewed version

Citation for published version (Harvard):
Minku, L 2021, Multi-stream online transfer learning for software effort estimation: is it necessary? in S McIntosh, X Xia & S Amasaki (eds), *PROMISE 2021: Proceedings of the 17th ACM International Conference on Predictive Models and Data Analytics in Software Engineering.* PROMISE: Predictor Models in Software Engineering, Association for Computing Machinery (ACM), pp. 11-20, The 17th International Conference on Predictive Models and Data Analytics in Software Engineering, Athens, Greece, 19/08/21.
https://doi.org/10.1145/3475960.3475988

Multi-stream Online Transfer Learning for Software Effort Estimation: Is It Necessary?

Leandro L. Minku
l.l.minku@cs.bham.ac.uk
School of Computer Science, The University of Birmingham
Birmingham, UK

ABSTRACT

Software Effort Estimation (SEE) may suffer from changes in the relationship between features describing software projects and their required effort over time, hindering predictive performance of machine learning models. To cope with that, most machine learning-based SEE approaches rely on receiving a large number of Within-Company (WC) projects for training over time, being prohibitively expensive. The approach Dycom reduces the number of required WC training projects by transferring knowledge from Cross-Company (CC) projects. However, it assumes that CC projects have no chronology and are entirely available before WC projects start being estimated. Given the importance of taking chronology into account to cope with changes, it may be beneficial to also take the chronology of CC projects into account. This paper thus investigates whether and under what circumstances treating CC projects as multiple data streams to be learned over time may be useful for improving SEE. For that, an extension of Dycom called OATES is proposed to enable multi-stream online learning, so that both incoming WC and CC data streams can be learnt over time. OATES is then compared against Dycom and five other approaches on a case study using four different scenarios derived from the ISBSG Repository. The results show that OATES improved predictive performance over the state-of-the-art when the number of CC projects available beforehand was small. Learning CC projects over time as multiple data streams is thus recommended for improving SEE in such scenario. When the number of CC projects available beforehand was large, OATES obtained similar predictive performance to the state-of-the-art. Therefore, CC data streams are unnecessary in this scenario, but are not detrimental either.

CCS CONCEPTS

• **Software and its engineering** → **Software development process management**; • **Computing methodologies** → *Online learning settings*; *Ensemble methods.*

KEYWORDS

Software effort estimation, cross-company learning, concept drift, transfer learning, data streams, ensembles

1 INTRODUCTION

Software Effort Estimation (SEE) consists in estimating the effort required to develop software projects (e.g., in person-hours), based on project features such as estimated size, team expertise, programming language, etc. Due to the difficulty in performing human-made effort estimations [15, 18], many researchers have investigated the possibility of using machine learning approaches as decision support tools for this task [8, 12, 19, 47]. However, building well performing machine learning SEE models is itself not an easy task. One of the challenges is that the training sets that a given company can collect from within its environments are typically small and heterogeneous, which can hinder predictive performance [10].

The relatively small training set size compared to other machine learning problems [25, 31] comes from the high cost of collecting the actual effort of software projects [21]. To overcome this problem, existing work has investigated the adoption of Cross-Company (CC) projects for training [19, 28]. CC data sets can alleviate the problem of small training set size because, even if each company acquires just a small number of training projects, a dataset containing data from several of such companies will be larger. Existing CC project repositories are available [17, 31] for SEE and some existing CC SEE approaches adopting them were able to maintain or improve predictive performance over WC SEE [38, 40, 48].

Heterogeneity means that the training sets are composed of a variety of software projects that are considerably different from each other. Among others, heterogeneity may occur as a result of different projects adopting widely different practices, or employing staff with considerably different backgrounds. This is the case both for Within-Company (WC) and Cross-Company (CC) datasets, but the use of CC training projects could potentially intensify heterogeneity [32]. Heterogeneity can also occur due to changes suffered by companies over time [38], such as key employees leaving the company, or changes in management strategies. These changes can alter the relationship between input features describing the projects and their required effort, a phenomenon called concept drift [13].

Local learning is a popular approach to tackle heterogeneity [6, 30, 39]. It gives predictions based only on the existing training projects that are most similar to the projects being estimated, potentially reducing the negative impact of heterogeneity. However, such locality is defined based on input features describing the software projects, being unable to deal with heterogeneity resulting from concept drift. To deal with concept drift, it is necessary to take into account the chronology of software projects, i.e., projects should be processed and learned as data streams. However, most SEE approaches for tackling concept drift (e.g., sliding window approaches [1–4, 20, 26, 27, 29]) rely on receiving a large number of WC projects for training over time. This makes them prohibitively expensive due to the high cost of collecting the required effort of such projects. The approach Dycom [40] reduces the number of required WC training projects by transferring knowledge from CC projects. However, it assumes that CC projects have no chronology and are all available before WC estimations start being required. Only the WC training projects form a data stream that can be learnt

over time. Given the benefits of learning WC projects over time to improve predictive performance by tackling concept drift, it may also be beneficial to learn CC training projects as data streams.

This paper thus aims at investigating whether and under what circumstances treating learning CC projects as multiple data streams to be learned over time may be useful for improving SEE. For that, an extension of Dycom called **O**nline Multi-Stre**a**m **T**ransfer **E**ffort E**s**timator (OATES) is proposed. Different from existing work, OATES is able to reduce the number of WC projects required for training without hindering predictive performance by learning both WC and CC training projects one-by-one as they arrive, i.e., in an online way. It overcomes Dycom's key limitation of requiring all CC training projects to be available beforehand. The following research questions are addressed:

- RQ1 Can OATES reduce the number of required WC training projects in SEE without hindering predictive performance? To what extent? This first question validates OATES by investigating whether it is able to maintain Dycom's strength and overcome sliding windows' weakness when WC training sets are small. This would mean that a company using OATES could potentially save the cost of collecting a large amount of WC training projects.
- RQ2 Can the use of CC data streams for training over time lead to improvements in predictive performance? Could it hinder predictive performance? To what extent and under what circumstances? This is the main question investigated in this paper. It determines whether it is useful to learn incoming CC projects over time through OATES.
- RQ3 When using incoming CC projects for training, to what extent collecting larger numbers of WC projects over time is still useful for SEE? The adoption of CC training projects is typically intended at reducing the need for collecting WC projects. However, it would be valuable to know whether acquiring larger numbers of WC training projects could further improve predictive performance when used in addition to CC training data. How beneficial that would be depends on the potential improvements in predictive performance that could be obtained. This question investigates that.

These RQs are answered based on a case study with projects from the International Software Benchmarking Standards Group (ISBSG) Repository [17]. OATES is compared against six approaches on four different scenarios. It was able to drastically reduce the number of WC projects required for training while maintaining or improving predictive performance, being a valid approach to reduce the cost of collecting WC projects while dealing with concept drift in SEE (RQ1). Learning CC training projects over time can be beneficial, but such benefit was only observed when the number of CC training projects available beforehand was limited. When that was not the case, learning additional CC training projects over time was not helpful, but was not detrimental either. These results suggest that it may not be necessary to cope with concept drift in the CC projects. Instead, these projects are only needed to maintain diverse knowledge that can be helpful for coping with concept drift in the WC projects (RQ2). In addition, when learning incoming CC projects over time, collecting larger numbers of WC training projects over time becomes unlikely necessary, as this rarely led to improvements in predictive performance (RQ3).

The remaining of this paper is organised as follows. Section 2 explains related work. Section 3 explains the proposed approach OATES. Section 4 explains the datasets used in the case study. Section 5 explains the experimental setup. Section 6 analyses the experiments and answers RQ1 to RQ3. Section 7 presents threats to validity. Section 8 discusses implications to practice. Section 9 presents conclusions and future work.

2 RELATED WORK

This section discusses existing work on CC SEE, sliding window approaches for SEE and time-transfer approaches for SEE, which are closely related to this study. For a brief general overview of machine learning for SEE, please refer to the supplementary material [35].

2.1 CC SEE

Collecting the actual effort of WC projects is expensive and lack of training examples can lead to poor predictive performance [10]. Therefore, several studies have tried to use CC projects to increase the size of the SEE training sets. However, simply adding any CC project to the training set has led to similar or worse performance than WC models [9, 19, 24, 52]. This is probably linked to the potential increase in the level of heterogeneity of datasets when using CC training projects [32]. Approaches that attempt to handle heterogeneity have shown more successful results in improving predictive performance over WC approaches [11, 23, 48, 49]. However, these approaches assume that heterogeneity only affects input features, lacking mechanisms to cope with concept drift.

2.2 Sliding Window (SL) SEE Approaches

Taking the chronology of projects into account acts as an enabler to tackle concept drift, given that concept drifts are changes observed over time in the relationship between input features and required effort of software projects. For instance, Kitchenham et al. [20] reported that the best fitting regression model changed substantially over time in a case study with a WC dataset. Premraj et al. [42] also reported that productivity changed over time in a study with a CC dataset. Therefore, if chronology is ignored, obsolete SEE models could lead to poor predictive performance.

SLs were the first machine learning SEE approaches specifically designed to cope with concept drift. One of the earliest examples is Kitchenham et al. [20]'s. For each new project p_t to be estimated, this approach trains a SEE model based on a "window" containing past WC projects p_{t-1} to p_{t-n}, where $n > 1$ is a pre-defined window size that specifies the number of past WC projects that the window can hold. Therefore, this window can be seen as "sliding" over the data stream formed by the WC projects received over time, respecting chronology and always keeping the same number of most recent projects. A good window size prevents training SEE models with old projects that could potentially represent different contexts[1], whereas poor choices can lead to worse predictive performance than models trained on all past projects [26].

Overall, studies investigated SLs with different datasets (CSC [20], ISBSG [1, 2, 26], MacDonell [29], Finnish [3, 27], Maxwell

[1]The term context is used in this work to refer to the relationship between input features and required effort in a company.

[4]) and machine learning algorithms (different variations of linear regression [2, 20, 26] and k-nearest neighbours [1, 3, 4]). The results suggest that SLs are more effective when using linear regression than k-nearest neighbours. Different variations of SLs have also been investigated, such as SLs whose size is determined by the duration (time) covered by the projects rather than by a fixed number of projects, and SLs that give higher weights for more recent projects. However, no evidence has been found in favour of duration-based SLs over SLs based on the number of projects, and weighting projects was only helpful when the windows were large.

SLs can be a simple yet effective way to handle concept drift for some companies, but they require a large number of WC training projects. This is because SLs require a good number of recent enough training projects so that they can cope with concept drift while still having enough training projects to produce well performing SEE models [3, 10, 26, 27].

2.3 Time Transfer SEE Approaches

Given the limitation of sliding windows when WC training sets are small, Minku and Yao [38, 41] investigated the use of CC projects to augment the training set while taking chronology and concept drift into account. Their approach was able to successfully transfer knowledge from past models to improve predictive performance over WC SEE models. This was achieved by tracking which WC and CC models were beneficial over time. However, when old SEE models are not beneficial, this approach still requires a large number of recent WC training projects to perform well.

Kocaguneli et al. [23] investigated a tree-based filtering approach called TEAK [22] to tackle heterogeneity. CC or WC training projects corresponding to sub-trees of high effort variance are assumed to be detrimental and filtered out. This approach managed to obtain similar performance when using only training projects from the same time period as the project being estimated, and when using training projects from a mixed time periods. However, it still relies on the availability of a good number of training projects from the same time period to generate the SEE model.

The approach Dycom [40] was proposed to deal with the problem of small WC training sets while still being able to tackle concept drift. It dynamically maps predictions given by past CC models to the current WC context, so that CC knowledge can be transferred even when such models do not match the WC context directly. Dycom was shown to reduce the number of WC projects required for training while maintaining or slightly improving predictive performance compared to WC approaches. Therefore, companies using it would be able to save the high cost of collecting a large number of WC training projects. However, Dycom only works for scenarios where enough CC projects are available beforehand to train the CC models. It is unable to deal with the dynamism of CC projects, i.e., with the fact that new CC training projects may arrive over time and suffer concept drift.

The proposed approach OATES is designed to overcome Dycom [40]'s limitation of not processing CC data streams, while keeping its advantage of requiring less WC projects than the other time transfer approaches and SL approaches. This will enable the investigation of the potential benefit of learning CC data streams.

3 PROPOSED APPROACH – OATES

This section proposes an extension of Dycom to enable CC projects to be learned over time. This approach is called **O**nline Multi-Stre**a**m **T**ransfer **E**ffort E**s**timator (OATES). Different from Dycom, it considers that not only WC, but also CC training projects arrive one-by-one in chronological order, i.e., in an online way. In machine learning, it is said that such projects form data streams. So, OATES is able to learn multiple (WC and CC) streams. Following [32], the term CC is used for projects believed to be potentially heterogeneous with respect to the projects being estimated. For example, projects from different departments of the same company could be considered as CC projects if such departments employ largely different practices.

As with Dycom, OATES contains the following components: a WC SEE model $\hat{f}_0 : \mathcal{X} \rightarrow \mathcal{Y}$, where $\mathcal{X}$ represents the space of input features describing software projects and $\mathcal{Y}$ represents the space of software required efforts; M CC SEE models $\hat{g}_i : \mathcal{X} \rightarrow \mathcal{Y}$, $1 \leq i \leq M$; M mapping functions $\hat{f}_i : \mathcal{Y} \rightarrow \mathcal{Y}$ of the format $\hat{f}_i(\hat{y}_i) = \hat{y}_i \cdot b_i$, $1 \leq i \leq M$, $b_i \in \mathbb{R}$, to map predictions $\hat{y}_i$ given by each CC SEE model $\hat{g}_i$ to the WC context; and $M + 1$ weights w_j, $0 \leq j \leq M$, to represent how helpful each SEE model $\hat{f}_j$ is likely to be for the WC context. The mapping functions together with their corresponding CC SEE models form mapped models $\hat{f}_i(\hat{g}_i) : \mathcal{X} \rightarrow \mathcal{Y}$ that can give predictions in the WC context. The $M + 1$ weights are thus associated to each mapped model and to the WC model. The effort estimation $\hat{f}(\mathbf{x})$ given to a project described by input features $\mathbf{x}$ is based on the weighted average of the predictions given by the WC model and the mapped models:

$$\hat{f}(\mathbf{x}) = \left[\sum_{i=1}^{M} w_i \cdot \hat{f}_i(\hat{g}_i(\mathbf{x}))\right] + w_0 \cdot \hat{f}_0(\mathbf{x}),$$

where $w_i > 0, \forall i \in \{0, 1, \cdots M\}$ and $\sum_{i=0}^{M} w_i = 1$. So, similar to Dycom, OATES uses an ensemble of mapped and WC SEE models.

However, OATES' procedures for training the CC SEE models, mapping functions and weights are different from those of Dycom, so that they can learn CC data streams instead of assuming that all CC data are available beforehand. To support that, OATES maintains an additional component that does not exist in Dycom: a sliding window W containing the most recent WC training projects. As OATES learns / updates its components over time not only based on WC, but also on CC data streams, this window is necessary to enable tracking changes that may affect the suitability of CC models to the WC context, including concept drifts affecting the CC data stream. By using this sliding window *as a means to transfer knowledge* from CC data streams, OATES overcomes both Dycom's weakness of requiring all CC data to be pre-available, and SL techniques' weakness of requiring a large number of WC training projects.

OATES' pseudocode is shown in Algorithms 1, 2 and 3. Lines in blue highlight steps that are different w.r.t. Dycom. The training procedure starts with initialisation (Algorithm 1, Line 1). The WC and CC models are initialised using the base machine learning algorithm to be adopted with OATES. This could be any existing supervised learning algorithm. The mapping functions associated to the CC models are initialised so that they perform a direct mapping between the CC SEE models and the WC context, i.e., $\hat{f}_i(\hat{g}_i(\mathbf{x})) = \hat{g}_i(\mathbf{x})$. The weights are initialised to zero. After that, the following components of OATES are trained:

Algorithm 1: OATES – Main Procedure. Parameters: β (factor for decreasing model weights); lr (learning rate for mapping function); M (number of CC models); ML: base machine learning algorithm

1 Initialise $\hat{f}_i$, $\hat{g}_j$, w_i and W, for $0 \leq i \leq M$ and $1 \leq j \leq M$
2 **for** *each new WC or CC training project* $(\mathbf{x}, y)$ **do**
3 **if** $(\mathbf{x}, y)$ *is a CC training project* **then**
4 Determine the CC model $\hat{g}_i$ to be trained on $(\mathbf{x}, y)$
5 Update $\hat{g}_i$ using ML and $(\mathbf{x}, y)$
6 Train mapping function $\hat{f}_i$ using W and lr (Alg. 2)
7 Calculate weights $w_i, 0 \leq i \leq M$, using W and β (Alg. 3)
8 **else**
 `// (x, y) is a WC training project`
9 Insert $(\mathbf{x}, y)$ into the end of the sliding window W
10 Remove the first project from W if $size(W) > 10$
11 Calculate weights $w_i, 0 \leq i \leq M$,using W and β (Alg. 3)
12 Train mapping functions $\hat{f}_i, 1 \leq i \leq M$, using W and lr (Alg. 2)
13 Update $\hat{f}_0$ using ML and $(\mathbf{x}, y)$

Algorithm 2: OATES – Learning Algorithm For Mapping Function $\hat{f}_i$. Parameters: lr (learning rate); W (sliding window of WC training projects); $\hat{g}_i$: CC model to be mapped

1 **if** $\hat{g}_i$*'s learning has already started* **then**
2 $b_i \leftarrow 1.0$
3 **for** *each project* $(\mathbf{x}, y) \in W$ **do**
4 **if** $(\mathbf{x}, y)$ *is the first project in* W **then**
5 $b_i \leftarrow y/\hat{f}_i(\mathbf{x})$
6 **else**
7 $b_i \leftarrow (1 - lr) \cdot b_i + lr \cdot y/\hat{f}_i(\mathbf{x})$

Algorithm 3: OATES – Weight Update Method. Parameters: β (factor for decreasing model weights); M (number of CC learners); W (window of WC training projects); $\hat{f}_i$, $0 \leq i \leq M$: SEE models

1 Set to 1.0 all weights w_i associated to models $\hat{f}_i, 0 \leq i \leq M$, whose training has already started
2 $errors \leftarrow \{\}$
3 **for** *each project* $(\mathbf{x}, y) \in W$ **do**
4 **for** *each* $\hat{f}_i, 0 \leq i \leq M$, *whose training has started* **do**
5 $errors \leftarrow errors \cup |y - \hat{f}_i(\mathbf{x})|$
6 Determine the winner model $\hat{f}_w$, which corresponds to the smallest error in $errors$
7 **for** *each* $\hat{f}_i, 0 \leq i \leq M \wedge i \neq w$ **do**
8 $w_i \leftarrow w_i \cdot \beta$
9 **for** *each* $\hat{f}_i, 0 \leq i \leq M$, *whose training has started* **do**
10 $w_i \leftarrow \text{normalise}(w_i)$

WC SEE model: similar to previous work [40, 41], whenever a new WC training project arrives, it is used to update a WC model $\hat{f}_0$ (Algorithm 1, Line 13), using a pre-defined (base) machine learning algorithm. Heterogeneity in the WC input features can be dealt with by using local base learning approaches [6, 30, 39].

CC SEE models: CC training projects are used to increase the number of training examples and potentially reduce overfitting. Whenever a new CC training project is received, OATES determines which CC SEE model should be trained with this project based on a clustering algorithm (Algorithm 1, Line 4) before performing such training (Algorithm 1, Line 5). Clustering injects extra locality in the learning procedure to help tackling the potentially higher heterogeneity of CC projects. The clustering algorithm should be able to operate in an online way and yet avoid past CC projects to switch between clusters, which would cause instability. Therefore, OATES uses a simple yet effective clustering strategy [36, 40]. It separates CC training projects based on productivity thresholds. For instance, if $M = 3$, $M - 1 = 2$ thresholds th_1 and th_2 are used. Projects with productivity $pr \leq th_1$, $th_1 < pr \leq th_2$ and $pr > th_2$ are associated to the first, second and third cluster and its corresponding CC model, respectively. Different from previous work [36, 40], training projects in OATES are clustered on the fly, as they arrive. Therefore, CC clusters can be associated to an increasing number of CC projects over time without causing past projects to switch between clusters.

Mapping functions: WC training projects are used to learn the mapping functions. However, as the CC models will be updated over time with the CC data stream, OATES' mapping function learning algorithm should consider not only potential concept drifts affecting the WC, but also the CC context. Therefore, OATES cannot use the same learning algorithm used by previous work [40]. Instead, OATES uses a sliding window W containing 10 WC training projects to learn the mapping function's internal parameter b_i from scratch whenever a new WC or CC training project is made available (Algorithm 1, Lines 6 and 12). The window is updated whenever a new WC training project is received (Algorithm 1, Lines 9 and 10). The reason for the window size of 10 is linked to the algorithm used to learn b_i, and will be explained below.

The factor b_i is initialised to 1.0, leading to a direct mapping between $\hat{f}_i$ and the WC context (Algorithm 2, Line 2). The window W is then used to learn b_i based on an exponential decay function. In particular, the first WC training project in W is used to create a perfect mapping between the prediction given by a CC model and the true effort of this WC project (Algorithm 2, Line 5) [40]. For all other WC training projects in W, an exponential decay function with smoothing factor lr is used to set b_i (Algorithm 2, Line 7). This is the weighted average of the value that would provide a perfect mapping for the current WC training project and the previous value of b_i, calculated based on the previous projects in W. A high smoothing factor lr puts more emphasis on the most recent WC training projects and achieves higher adaptability to concept drift, whereas a low lr leads to a more stable mapping function. Exponential decay means that older WC training projects become exponentially less important in comparison to the most recent WC training project. Therefore, window sizes larger than 10 would not have a significant impact on OATES' performance. Preliminary experiments with other window sizes confirmed that. In summary, the exponential decay function allows us to obtain good mapping functions based on previous WC training projects, while enabling adaptability to both WC and CC concept drifts.

Weights: OATES' weights are also calculated based on the sliding window W (Algorithm 1, Lines 7 and 11). The procedure starts with setting the initial weights (Algorithm 3, Line 1). After that, for each WC training project in W, the WC or mapped model which provided the lowest absolute error is considered to be the *winner* (Algorithm 3, Lines 3 to 6). All models except the winner have their weights multiplied by a pre-defined parameter β ($0 < \beta \leq 1$), and then all weights are normalised in order to sum to one (Algorithm 3, Lines 8 to 10). This is done by dividing the weights by the sum of the weights of all models $\hat{f}_i$ that have already started being trained.

4 DATASETS

Obtaining SEE datasets for CC data stream studies is challenging. To enable a proper analysis, datasets need to have: (1) projects from multiple companies, (2) use of the same input features for all projects, (3) information on which projects belong to a given single company, (4) chronology information. A dataset that satisfies all these requirements is ISBSG [17] Release 10, where information on (3) was provided upon request. Moreover, the singled out company from this dataset has a relatively large number of projects (187 in total; 184 after preprocessing) in comparison with some other SEE datasets [31]. By training OATES with a small portion of such WC projects and comparing its performance against WC approaches trained on the full WC training set, we can investigate how successful OATES is in tackling small WC training sets.

The projects were described by four input features: development type (enhancement, new development, re-development); language type (2GL, 3GL, 4GL, ApG), development platform (multi-platform, mainframe, PC, mid-range); and functional size (numeric). These input features are the ones suggested by ISBSG's guidelines as the most important criteria for estimation purposes [17], and were the same as those adopted in previous work [40]. The output feature is the software development effort in person-hours. The dataset was pre-processed as described in previous work [41].

Four datasets representing different scenarios were derived:

- ISBSG2000 – 168 CC projects implemented in the year interval [1993–2000], and 119 WC projects implemented in [2001–2003].
- ISBSG2001 – 224 CC projects implemented in [1993–2001], and 69 WC projects implemented in [2002–2003].
- ISBSG – 415 CC projects implemented in [1993–2003], and 184 WC projects implemented in [1996–2003]. This is the full WC ISBSG dataset containing all CC projects until the end of the period covered by the WC projects.
- ISBSGLess – 226 CC implemented in [1993–1994;2001–2003], and 90 WC projects implemented in [2001–2003].

ISBSG2000 and ISBSG2001 have all their CC projects implemented before WC estimations start being required. They were adopted in [40] to evaluate Dycom, and are used here to investigate whether OATES maintains the strengths of that approach. ISBSG and ISBSGLess were created for this study to evaluate OATES in two different scenarios where additional CC projects are available over time after WC estimations start being required. ISBSG contains a considerable number of CC projects (94) prior to the WC projects, whereas ISBSGLess contains much less (only 10).

5 EXPERIMENTAL SETUP

To answer the RQs, the following approaches are compared:

- OATES: this approach was given access to only one in every $P = \lfloor n/6 \rfloor$ WC projects for training, where n is the size of the WC dataset. This value was chosen so that the number of WC projects used for training is equal to the smallest number of WC training projects used in previous work with Dycom [40]. It means that only 6 WC projects are available for training, independent of the dataset. This value is used in all experiments to answer RQ1 and RQ2. Other values ($P \in \{\lfloor n/12 \rfloor, \lfloor n/30 \rfloor, \lfloor n/60 \rfloor\}$) will be used to analyse the impact of the number of WC training projects (RQ3).
- Dycom [40]: this is the state-of-the-art time transfer approach (Section 2.3). As with OATES, it has access to only one in every $P = \lfloor n/6 \rfloor$ WC projects for training. A comparison against Dycom enables us to check how helpful it is to learn incoming CC training projects over time, rather than only allowing pre-existing CC training projects (RQ2).
- SL $P = 1$ [26]: this is a typical SL approach (Section 2.2) where all (i.e., one in every $P = 1$) WC projects received so far are available for training. It uses a sliding window with a fixed number of WC projects. It enables us to check how successful OATES is in reducing the number of WC projects required for training (RQ1).
- WC $P = 1$: this is a WC model trained on all available WC projects so far ($P = 1$) using a given machine learning algorithm. Comparison against it complements the answer to RQ1.
- SL $P/6$: this is the same as SL $P = 1$, but using $P = \lfloor n/6 \rfloor$. Comparison against it enables us to check OATES' performance against other approaches using the same limited number of WC training projects (RQ1).
- WC $P/6$: this is the same as WC $P = 1$, but using $P = \lfloor n/6 \rfloor$. Comparing it complements the answer to RQ1.
- Median: this is a baseline that predicts the median of the efforts of all previously seen WC training projects. Similar to OATES, it uses $P = \lfloor n/6 \rfloor$. Reasonable SEE approaches should perform better than Median. Median was selected rather than mean because it can frequently provide better estimates [37], as its estimations are not affected by outlier projects. More complex baselines such as [43, 51] were not adopted because they are not prepared for processing data streams, which is required for this study. Moreover, they were implemented in R and thus cannot be integrated with the MOA [7] framework adopted in this study to enable them to process data streams. Linear regression and linear regression in the log scale as WP $P/6$ models will be used as alternatives to [51], which is also a linear regression model with transformations.

The following machine learning algorithms were used to train OATES', Dycom's and SL's base models, as well as the WC approaches: k-Nearest Neighbours, Regression Trees, Linear Regression and Linear Regression in the log scale. Euclidean distance over normalised input features was adopted by k-Nearest Neighbours. For categorical features, the difference between two values was set to 1 for different values, and 0 otherwise. The base learning algorithms were chosen for being competitive for SEE [12, 39, 51]. In particular, k-Nearest Neighbours and Regression Trees are local learning approaches, which try to cope with heterogeneity based on input features [30, 39, 46]. Linear Regression (including in the log scale) is expected to be competitive [12] due to the relative linearity of several SEE datasets. All algorithms are implemented in WEKA [16] and MOA [7], which is an open source framework for

running and evaluating data stream learning algorithms. OATES' source code is available at [34].

All approaches are requested to estimate all WC projects for testing. Each WC project is required to be estimated once, at a given *time step*, in chronological order. This reflects a real world scenario where a company would wish to provide an estimation for each of its projects. Similar to other machine learning data stream studies [14], after a given WC project is estimated, SL $P = 1$ and WC $P = 1$ assume that the true effort for this project will become known. This project can then be used for training. However, approaches with $P > 1$ assume that only one in every $P > 1$ WC projects have their true efforts revealed, being trained on less WC projects. No WC project is used for training before being used for testing.

The testing performance measures used in this work were the Mean Absolute Error (MAE) and the Mean Absolute Error in the log scale (MLogAE), calculated at each time step t over a sliding window containing n' past errors obtained when estimating WC projects over time:

$$MAE_t = \frac{1}{min(n',t)} \sum_{i=max(t-n'+1,1)}^{t} |\hat{y}_i - y_i|;$$
$$MLogAE_t = \frac{1}{min(n',t)} \sum_{i=max(t-n'+1,1)}^{t} |log(\hat{y}_i) - log(y_i)|,$$

where $\hat{y}_i$ is the estimation given to the WC project requested to be estimated at time step i, whose true effort is y_i. Measures based on the AE are unbiased towards under or overestimations, which is a desirable characteristic for evaluating SEE approaches [45]. MLogAE was used in addition to MAE because it is not biased towards large projects. The use of sliding windows for evaluation is recommended and widely used when data streams do not require memoriless and very fast evaluation [14]. It means that n measurements (MAE_t, $MLogAE_t$, $1 \leq t \leq n$) are taken for each performance measure for a WC dataset of size n. Each measurement reflects the recent predictive performance obtained by an approach, enabling us to track how this predictive performance changes over time. The window size was $n' = 10$, following previous work [40, 41].

To give an indication of the relative magnitude of the differences in performance, Standardised Accuracy (SA) was used:

$$SA = Round\left(\left[1 - \frac{AggMAE(App)}{AggMAE(Median)}\right] * 100\right),$$

where $AggMAE(App)$ and $AggMAE(Median)$ are the aggregation of the n MAE_t measurements taken for a given approach App and for the Median approach, respectively. The aggregation function Agg is the mean. This is the same as the SA measure recommended by Shepperd and McDonell [45] for SEE studies, but gives an indication of how much better an approach does in comparison to the Median baseline rather than random guessing. Median was favoured over random guessing because the latter converges to the Mean, which typically performs worse than Median for SEE [37]. Therefore, determining how much an approach outperforms Median is more informative for the purpose of this study.

Friedman tests, Nemenyi post-hoc tests and A12 effect sizes [5] were used to support the analyses. They were chosen for being non-parametric and appropriate for comparing multiple groups. Friedman's null hypothesis is that all compared approaches perform similar in terms of a given performance measure across time steps on a given dataset. The alternative hypothesis is that at least one pair of approaches perform different from each other. Nemenyi tests were used to identify which pairs of approaches perform different. As suggested in [50], absolute values of A12 $\geq$ 0.56, 0.64 and 0.71 indicate small, medium and large effect size, respectively.

Base learning algorithms were investigated with 20 different parameter combinations each. For each dataset, the parameter combination that led to the best median of the n MAE_t measurements for WC $P = 1$ was selected to be used with all approaches. The same number of 20 parameter combinations was investigated with OATES, Dycom and SL $P = 1$. For SL $P/6$, the maximum possible number of parameter choices is 5. So, all possible parameter values were investigated for this approach. WC $P/6$ has no additional parameters besides those of the base learning algorithm. So, no further tuning was applied. For each dataset and approach, the parameters and base learners leading to the best median of the n MAE_t measurements were chosen for the analysis. The parameter combinations are available in the supplementary material [35].

OATES and Dycom were assigned the same productivity thresholds for clustering CC projets. As in previous work [40], these thresholds were chosen so as to divide the CC training projects available prior to estimations into three similar size chunks containing CC projects of low, medium and high productivity. Productivity was measured based on the CC projects' normalised level 1 productivity rate in hours per functional size unit, provided by ISBSG. The thresholds were: 5.60 and 13.00 for ISBSG2000, 6.00 and 14.00 for ISBSG2001, 4.5 and 10.5 for ISBSG and 3.0 and 8.0 for ISBSGLess.

6 EXPERIMENTAL RESULTS

Table 1 and Figs. 1 and 2 show the overall results of the experiments. Larger versions of the figures and additional information about effect sizes are in the supplementary material [35].

6.1 Preliminary Analysis

As explained by Shepperd and McDonell [45], new approaches performing similar to simple baselines can be deemed unsuccessful, given that the baseline could be easily adopted instead. OATES was always significantly better than the Median baseline, both in terms of MAE and MLogAE (Fig. 1). The effect sizes of the differences were large most of the time (last row of Table 1). Overall, OATES performed from 15% to 59% better than the Median baseline (SAs in Table 1). Therefore, it is worth investigating OATES further.

6.2 RQ1: OATES Validation

RQ1 asks whether and to what extent OATES can reduce the number of required WC training projects without hindering predictive performance, validating OATES as an approach able to preserve Dycom's core advantages. Fig. 1 shows that OATES performed always at least as well as SL $P = 1$ and WC $P = 1$. Therefore, OATES can drastically reduce the number of WC projects needed for training without hindering MAE and MLogAE. This is a very positive result, as companies using OATES could potentially drastically reduce the cost of collecting WC training projects.

In addition, OATES' performance was sometimes significantly better than that of SL $P = 1$ and WC $P = 1$. Its MAE was significantly better than that of SL $P = 1$ for ISBSG2000, ISBSG2001 and ISBSG, and than that of WC $P = 1$ for ISBSG2000 and ISBSG2001. Differences in SA, which is a measure derived from MAE, varied from 1% to 10% when there was significant difference in MAE (Table 1). OATES' MLogAE was significantly better than that of WC $P = 1$

Table 1: Overall Performance Across Time Steps

Approach	ISBSG2000			ISBSG2001			ISBSG			ISBSGLess		
	AggMAE	AggMLogAE	SA	AggMAE	AggMLogAE	SA	AggMAE	AggMLogAE	SA	AggMAE	AggMLogAE	SA
OATES	2308.45	0.3284	21%	2439.18	0.3768	59 %	2550.02	0.3134	26 %	2385.71	0.3999	15%
Dycom	2357.76	*0.3625	19%	2568.90	*0.4032	57 %	2705.56	0.3219	22 %	*2655.37	**0.4800	6%
SL $P = 1$	*2494.59	***0.5608	15 %	*2942.41	**0.6051	51 %	*2610.17	***0.4870	25%	2389.83	0.4550	15%
WC $P = 1$	2385.61	***0.5491	18 %	***3057.94	***0.5902	49 %	2741.54	0.3738	21 %	*2000.31	0.4823	29%
SL $P/6$	**2922.67	***0.5111	0%	***4064.76	***0.6718	32 %	**3420.80	***0.4535	1%	*2682.08	***0.5677	5%
WC $P/6$	**2914.94	***0.5100	0%	***5159.18	***0.7312	14 %	**3444.57	***0.4580	0%	*2689.17	***0.5691	4%
Median	**2923.18	***0.5118	N/A	***5984.91	***0.7911	N/A	**3461.36	***0.4604	N/A	*2815.31	***0.6204	N/A

Cells in lime (light grey) correspond to the top ranked approach and to approaches whose performance was not significantly different from the top ranked approach according to the Nemenyi tests shown in Fig. 1. Cells with *, ** and *** indicate small, medium and large effect size w.r.t. OATES, respectively.

for ISBSG2000, ISBSG2001 and ISBSG, and than that of WC $P = 1$ for ISBSG2000, ISBSG2001 and ISBSGLess. OATES' MLogAE was from 17% to 41% smaller in these cases. Several of the effect sizes for the differences were large (Table 1). Therefore, OATES can not only save the cost of collecting a large number of WC training projects, but also sometimes considerably improve predictive performance.

When analysing online learning approaches, it is also important to check the performance over time. This is because, even though the overall performance for a given approach may be similar or better, there could be periods of time when its performance was worse. From Fig. 2, we can see that OATES' performance was quite consistently similar to or better than that of SL $P = 1$ and WC $P = 1$, except for ISBSGLess in terms of MAE, where OATES was a bit worse than these approaches during a considerable period of time. There were some periods of time when OATES performed worse than these approaches in terms of MLogAE for ISBSGLess as well, but this is justified by the higher stability that OATES presented for this dataset. This demonstrates that OATES behaves well overall, even though there is still some room for improvement. In particular, OATES behaved better in terms of MLogAE than MAE for ISBSGLess. This is corroborated by the different ranks in Fig. 1. It indicates that the WC approaches tended to do better for unusually large WC projects in the beginning of this dataset, whereas OATES tended to do better for a larger number of more typical WC projects during this period. The better behaviour achieved by SL $P = 1$ and WC $P = 1$ for larger projects may be due to the larger number of such unusual projects that they had access to for training.

SL $P/6$ and WC $P/6$ typically obtained much worse results than OATES (Table 1). The differences in MAE and MLogAE were always significant (Fig. 1). The differences in SA varied from 10 to 45%, and OATES' MLogAE was from 30% to 48% smaller (Table 1). The effect sizes were frequently large (Table 1). This demonstrates a considerably large improvement in predictive performance achieved by OATES. It confirms that SL and WC are not well prepared to deal with small WC training sets as explained in Section 2, and that approaches such as OATES can help to overcome this issue.

RQ1: OATES managed to use CC projects to drastically reduce the number of WC training examples used by SL $P = 1$ and WC $P = 1$, while maintaining or improving predictive performance (differences in SA of up to 10% and MLogAEs up to 41% smaller). The results suggest that OATES performed particularly well on more typical projects, but sometimes struggled on unusually large ones.

6.3 RQ2: The Benefit of Incoming CC Projects

RQ2 asks whether and to what extent CC training projects that arrive over time can help or hinder predictive performance. For that, we compare OATES against Dycom, which is able to use CC training projects, but only those available before WC estimations are required. From Fig. 1, we first observe that OATES performed similar to Dycom on ISBSG2000 and ISBSG2001. These datasets did not provide additional CC training projects over time. This further confirms that OATES was successful in maintaining Dycom's strength of benefiting from CC training projects to reduce the number of required WC training projects.

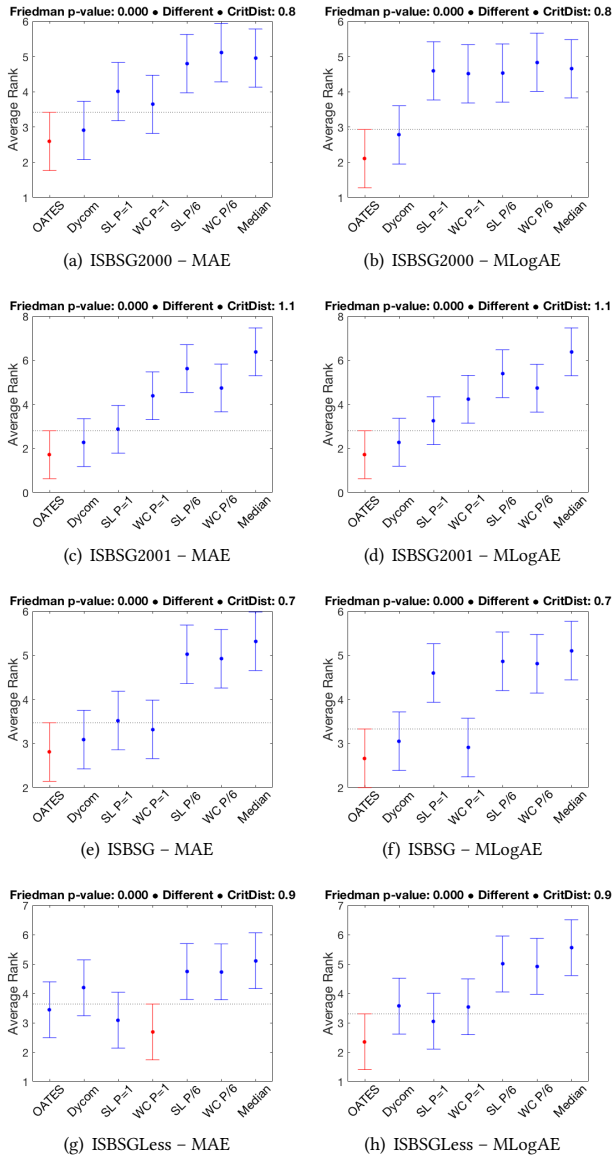
OATES and Dycom also achieved similar predictive performance for ISBSG (Fig. 1). This was the dataset where the number of CC training projects available beforehand was already large, despite additional CC training projects becoming available over time. Therefore, in this scenario, learning additional CC training projects was not beneficial, but did not hinder predictive performance either. It is likely that the large number of pre-existing CC projects were enough for achieving good improvements in predictive performance for the whole period analysed. It means that old training projects can potentially be useful for prolonged periods of time.

For ISBSGLess, OATES performed similar to Dycom in terms of MAE, but better in terms of MLogAE, with medium effect size (Fig. 1 and Table 1). OATES' MLogAE was 17% smaller than that of Dycom (Table 1), representing a considerable difference. These results indicate that OATES' mechanisms can help to make use of the incoming CC training projects to make up for the lack of CC projects available prior to the WC estimates in this dataset. The differences of results in terms of MAE and MLogAE suggest that there is a large number of more typical projects for which OATES performed better than Dycom, even though there is a small number of unusually large projects which cause these approaches to perform similarly in terms of MAE.

Fig. 2 shows that OATES' performance is indeed similar to Dycom's most of the time, except ISBSGLess in terms of MLogAE, where OATES considerably outperforms Dycom for extended periods of time. This corroborates the analyses above.

It is worth noting that, when no CC projects are available beforehand, Dycom is equivalent to WC $P/6$. From Section 6.2, we can see that Dycom (which would be equivalent to WC $P/6$ in Table 1 and Figs. 1 and 2) always performed very poorly when no CC projects were available beforehand. Therefore, the ability to learn CC data streams is essential when there are no pre-existing CC projects.

It is rather surprising that processing CC projects as data streams was only helpful when the number of pre-existing CC projects was limited. This suggests that it might not be necessary to cope with concept drifts that may affect the CC models. Instead, these models may be used to represent a diversity of software projects that may



(a) ISBSG2000 – MAE (b) ISBSG2000 – MLogAE

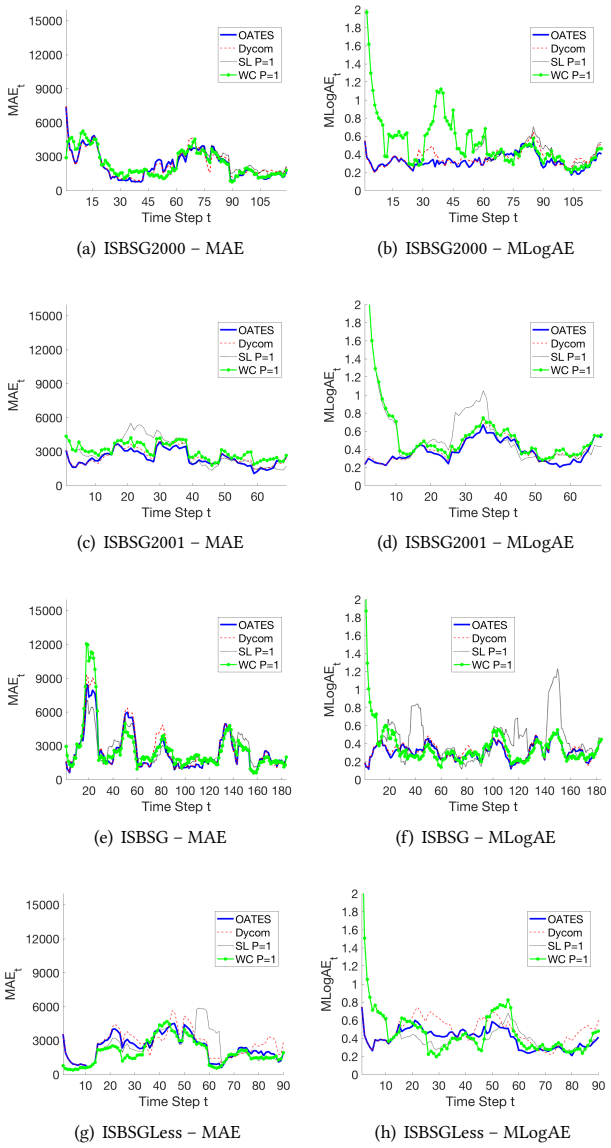
(c) ISBSG2001 – MAE (d) ISBSG2001 – MLogAE

(e) ISBSG – MAE (f) ISBSG – MLogAE

(g) ISBSGLess – MAE (h) ISBSGLess – MLogAE

The top ranked approach is shown in red. The dotted horizontal line represents Nemenyi's critical distance with respect to the mean rank of the top ranked approach. Approaches whose mean rank is above this line are significantly different from the top ranked approach.

Figure 1: Friedman and Nemenyi Tests Comparing Different Approaches in Terms of MAE and MLogAE Across Time.



(a) ISBSG2000 – MAE (b) ISBSG2000 – MLogAE

(c) ISBSG2001 – MAE (d) ISBSG2001 – MLogAE

(e) ISBSG – MAE (f) ISBSG – MLogAE

(g) ISBSGLess – MAE (h) ISBSGLess – MLogAE

Figure 2: MAE and MLogAE Across Time.

be useful for dealing with concept drift in the WC data. Therefore, dealing with concept drift in the WC data may be enough.

RQ2: Learning additional CC projects that arrive over time helped to improve MLogAE by 17% when the number of CC projects available beforehand was small. For scenarios where additional CC projects were not helpful, they were not detrimental either.

6.4 RQ3: Impact of the Number of WC Projects

RQ3 investigates how predictive performance is affected by the number of WC projects available for training when using CC data streams. To answer RQ3, Friedman and Nemenyi tests were performed to compare OATES' MAE and MLogAE when varying the number of WC training projects from smaller ($P = \lfloor n/6 \rfloor$, denoted by $P/6$) to higher ($P = \lfloor n/60 \rfloor$, denoted by $P/60$). Plots with graphical representations of the results were omitted due to space constraints, but are available in the supplementary material [35].

Increasing the number of WC training projects did not always lead to improvements in predictive performance. For ISBSG2000 and ISBSG2001 in terms of MAE, Friedman tests found no significant difference among the different numbers of WC training projects (p-values of 0.268 and 0.374, respectively). For ISBSGLess, there was no significant difference in terms of MLogAE (p-value of 0.053).

For ISBSG2000's MLogAE, ISBSG2001's MLogAE, ISBSG's MAE and MLogAE and ISBSGLess MAE, Friedman found significant differences (p-values of 0.038 for ISBSG2000 and <0.001 for the

others). However, the Nemenyi tests reveal that these differences are either in favour of adopting $P/6$ (for ISBSG's MLogAE) or there were no significant differences between $P/6$ and the best P value (all other cases except for ISBSGLess' MAE). The only case where $P/6$ led to worse predictive performance than higher numbers of WC training projects was ISBSGLess in terms of MAE, but the effect size was basely small (0.56).

It is worth noting in particular that increasing P from $P/6$ to $P/60$ was not helpful. Either there were no significant differences between these values of P (for ISBSG2000 and ISBSG2001 in terms of MLogAE and ISBSG and ISBSGLess in terms of MAE) or there were differences in favour of $P/6$ (ISBSG in terms of MLogAE). This is different from the behaviour of SL and WC, where using the full dataset often led to better results than $P/6$ (Fig. 1). A possible reason for that is that OATES' mechanisms have been designed to reduce the number of required WC training projects. Using a larger number of WC training projects may cause OATES to concentrate too much on too recent projects, without taking enough history into account. This is because the methods for updating the mapping function and the weights of the base learners are based on exponential decay. So, if the WC training projects added to the end of the window are all very recent, only the very recent period of time will be considered.

RQ3: Larger numbers of WC training projects did not lead to reasonable improvements in predictive performance when using CC data streams through OATES. This indicates that it is not worth collecting a large number of WC training projects for use with OATES.

7 THREATS TO VALIDITY

Internal validity: poor parameter choices can highly influence machine learning results. All approaches (including existing ones from the literature) except for SL $P/6$ and WC $P/6$ were tuned by investigating 20 parameter configurations each as explained in Section 5. This prevents approaches with more parameters from inappropriately receiving a higher level of parameter tuning attention. SL $P/6$ and WC $P/6$ are exceptions to this rule for the following reason. SL $P/6$'s possible parameter choices are very limited – just 5 possible choices. WC $P/6$ has no further parameters to be tuned besides those of the base learning algorithm. Whilst a smaller number of parameter configurations might be seen as a potential advantage for some approaches, it does not translate into an advantage for SL $P/6$ and WC $P/6$. This is due to their inability to perform well with a limited number of WC training examples. In practice, choosing parameters for any approach operating in online learning scenarios is not straightforward, because no WC data is available beforehand to create separate validation sets. Even though some research effort exists in this area [33], more efficient approaches to automatically tune parameters over time are still desirable.

Construct and statistical conclusion validity: the analyses were mainly based on MAE, MLogAE and SA. As explained in Section 5, they have been chosen following advice from the literature on the use of unbiased performance measures [45] and on data stream performance evaluation [14]. They include a measure that is influenced by project size, a measure that is not influenced by project size, and a derived measure that allows for better interpretation of the magnitudes of the differences in performance. WC projects were never used for training before having been used for testing. Non-parametric Friedman and Nemenyi tests, as well as A12 effect size, were used to further address conclusion validity.

External validity: this study is based on ISBSG datasets, which have the characteristics necessary to conduct the analyses, as explained in Section 4. Even though they have data overlaps, these datasets represent different scenarios. In particular, online learning can behave very differently based on which examples are available before a given time step [38, 40, 41]. This was also observed in Sections 6.2 and 6.3, i.e., results differed for different scenarios. Therefore, using different scenarios contributes towards generalisability when a single repository is available for the experiments, even when there are data overlaps. However, results may not generalise to other datasets. Future investigation with other datasets will be valuable once appropriate datasets become available.

8 IMPLICATIONS TO PRACTICE

The results obtained by SEE models have been considerably improving over the years [44]. However, due to the difficulty of this task, both machine learning models and human experts still make mistakes in their estimations. They could thus be used to complement each other. For instance, if their estimations are similar, we have an increased confidence on the estimate. If their estimations differ considerably, the expert may wish to analyse the project further to gain more insights into what effort best reflects the project. As OATES can learn over time and can adapt to concept drifts, it could potentially help to overcome difficulties that humans have in adapting and improving their estimations over time [15].

By enabling CC training projects to be learnt over time, OATES has the potential to save the cost of companies collecting large numbers of WC training projects for creating SEE models while enabling models to be updated to consider potentially new technologies and software development processes. Organisations worldwide provide increasing numbers of CC projects through repositories such as ISBSG [17] which could be used for that. A company adopting OATES would need to ensure that the input features used to describe their WC projects are the same as those used by the CC projects, and put procedures in place to acquire CC training projects in addition to a small number of WP training projects.

OATES could also be potentially used to provide a better understanding of the relationship between the context of different companies over time by visualising the mapping function. This could inform strategies to improve a company's productivity, or to track improvements over time once a company adopts a given strategy. This will be investigated as future work.

9 CONCLUSIONS

This paper provided the first investigation of whether and under what circumstances it is worth adopting CC multi-stream transfer learning for SEE. For that, an approach called OATES was proposed. A case study with the ISBSG Repository shows that OATES was able to drastically reduce the number of WC projects required for training, while maintaining and sometimes even considerably improving predictive performance, being a valid CC learning approach (RQ1). Learning CC data streams through OATES led to improvements in MLogAE when the number of CC training projects available before the WC projects was small. When this number was large, additional CC training projects did not help, but were not detrimental either

(RQ2). Finally, no evidence has been found in favour of collecting larger numbers of WC training projects when CC training projects are learned over time through OATES (RQ3).

Future work includes investigating OATES with other datasets (when these become available) and clustering algorithms; investigating how OATES could provide insights into the relationship between the context of different companies over time; analysing its computational time; and proposing novel online approaches for automatically tuning parameters.

ACKNOWLEDGMENTS

This work was supported by EPSRC Grant No. EP/R006660/2.

REFERENCES

[1] S. Amasaki and C. Lokan. 2012. The Effects of Moving Windows to Software Estimation: Comparative Study on Linear Regression and Estimation by Analogy. In *IWSM*. Assisi, Italy, 23–32.

[2] S. Amasaki and C. Lokan. 2015. On the effectiveness of weighted moving windows: Experiment on linear regression based software effort estimation. *Journal of Software: Evolution and Process* 27, 7 (2015), 488–507.

[3] S. Amasaki and C. Lokan. 2015. A Replication of Comparative Study of Moving Windows on Linear Regression and Estimation by Analogy. In *PROMISE*. 6.1–6.10.

[4] S. Amasaki, Y. Takahara, and T. Yokogawa. 2011. Performance Evaluation of Windowing Approach on Effort Estimation by Analogy. In *IWSM*. Nara, 188–195.

[5] A. Arcuri and L. Briand. 2011. A Practical Guide for using statistical tests to assess randomized algorithms in software engineering. In *ICSE*. Waikiki, 1–10.

[6] N. Bettenburg, M. Nagappan, and A. E. Hassan. 2012. Think Locally, Act Globally: Improving defect and effort prediction models.. In *MSR*. Zurich, 60–69.

[7] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer. 2010. MOA: Massive Online Analysis. *JMLR* 11 (2010), 1601–1604.

[8] B. Boehm. 1981. *Software Engineering Economics*. Prentice-Hall, NJ.

[9] L.C. Briand, T. Langley, and I. Wieczorek. 2000. A Replicated Assessment of Common Software Cost Estimation Techniques. In *ICSE*. Como, Italy, 377–386.

[10] V. S. Cherkassky and F. Mulier. 1998. *Learning from Data: Concepts, Theory, and Methods*. John Wiley & Sons, Inc., New York.

[11] A. Corazza, S. Di Martino, F. Ferrucci, C. Gravino, and F. Sarro. 2015. From Function Points to COSMIC - A Transfer Learning Approach for Effort Estimation. In *PROFES*. Bolzano, 251–267.

[12] K. Dejaeger, W. Verbeke, D. Martens, and B. Baesens. 2012. Data Mining Techniques for Software Effort Estimation: A comparative study. *IEEE TSE* 38, 2 (2012), 375–397.

[13] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar. 2015. Learning in Nonstationary Environments: A Survey. *IEEE CIM* 10, 4 (2015), 12–25.

[14] J. Gama, R. Sebastiao, and P. P. Rodrigues. 2009. Issues in Evaluation of Stream Learning Algorithms. In *KDD*. Paris, 329–337.

[15] T. M. Gruschke and M. Jørgensen. 2008. The Role of Outcome Feedback in Improving the Uncertainty Assessment of Software Development Effort Estimates. *ACM TOSEM* 17, 4 (2008), 20:1–20:35.

[16] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. 2009. The WEKA Data Mining Software: An update. *SIGKDD Explorations* 11, 1 (2009), 10–18.

[17] ISBSG. 2011. The International Software Benchmarking Standards Group. http://www.isbsg.org

[18] M. Jørgensen and S. Grimstad. 2011. The Impact of Irrelevant and Misleading Information on Software Development Effort Estimates: A Randomized Controlled Field Experiment. *IEEE TSE* 37, 5 (2011), 695–707.

[19] B.A. Kitchenham, E. Mendes, and G.H. Travassos. 2007. Cross versus Within-Company Cost Estimation Studies: A Systematic Review. *IEEE TSE* 33, 5 (2007), 316–329.

[20] B. Kitchenham, S.L. Pfleeger, B. McColl, and S. Eagan. 2002. An Empirical Study of Maintenance and Development Estimation Accuracy. *JSS* 64, 1 (2002), 57–77.

[21] E. Kocaguneli, B. Cukic, T. Menzies, and H. Lu. 2013. Building a Second Opinion: learning cross-company data. In *Proceedings of the 9th International Conference on Predictive Models in Software Engineering (PROMISE)*. 12.1–10.

[22] E. Kocaguneli, T. Menzies, A. Bener, and J. W. Keung. 2012. Exploiting the Essential Assumptions of Analogy-Based Effort Estimation. *IEEE TSE* 38, 2 (2012), 425–438.

[23] E. Kocaguneli, T. Menzies, and E. Mendes. 2015. Transfer Learning in Effort Estimation. *EMSE* 20, 3 (2015), 813–843.

[24] M. Lefley and M. Shepperd. 2003. Using Genetic Programming to Improve Software Effort Estimation Based on General Data Sets. In *GECCO*, Vol. LNCS 2724. Chicago, 2477–2487.

[25] M. Lichman. 2013. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml

[26] C. Lokan and E. Mendes. 2009. Applying Moving Windows to Software Effort Estimation. In *ESEM*. Lake Buena Vista, Florida, USA, 111–122.

[27] C. Lokan and E. Mendes. 2014. Investigating the use of duration-based moving windows to improve software effort prediction: A replicated study. *IST* 56, 9 (2014), 1063–1075.

[28] S. MacDonell and M. Shepperd. 2007. Comparing Local and Global Software Effort Estimation Models – Reflections on a Systematic Review. In *METRICS*. Madrid, 401–409.

[29] S.G. MacDonell and M. Shepperd. 2010. Data Accumulation and Software Effort Prediction. In *ESEM*. Bolzano-Bolzen, Italy, 31.1–31.5.

[30] T. Menzies, A. Butcher, D. Cok, A. Marcus, L. Layman, F. Shull, B. Turhan, and T. Zimmermann. 2013. Local vs. Global Lessons for Defect Prediction and Effort Estimation. *IEEE TSE* 39, 6 (2013), 822–834.

[31] T. Menzies, R. Krishna, and D. Pryor. 2017. The SEACRAFT Repository of Empirical Software Engineering Data. tiny.cc/seacraft

[32] L. Minku. 2016. On the Terms Within- and Cross-Company in Software Effort Estimation. In *PROMISE*. Ciudad Real, Spain, 4.1–4.4.

[33] L. Minku. 2019. A Novel Online Supervised Hyperparameter Tuning Procedure Applied to Cross-Company Software Effort Estimation. *EMSE* 24 (2019), 3152–3204. Issue 5.

[34] L. Minku. 2021. minkull/OATES: (Version v1.0). Zenodo. http://doi.org/10.5281/zenodo.5068001

[35] L. Minku. 2021. Multi-stream Online Transfer Learning for Software Effort Estimation: Supplementary Material. https://www.cs.bham.ac.uk/~minkull/publications/MinkuPROMISE2021-supplement.pdf

[36] L. Minku and S. Hou. 2017. Clustering Dycom: An Online Cross-Company Software Effort Estimation Study. In *PROMISE*. Toronto, 12–21.

[37] L. Minku, F. Sarro, E. Mendes, and F. Ferrucci. 2015. How to Make Best Use of Cross-Company Data for Web Effort Estimation?. In *ESEM*. Bergamo, Italy.

[38] L.L. Minku and X. Yao. 2012. Can Cross-company Data Improve Performance in Software Effort Estimation?. In *PROMISE*. Lund, Sweden, 69–78.

[39] L.L. Minku and X. Yao. 2013. Ensembles and Locality: Insight on Improving Software Effort Estimation. *IST* 55, 8 (2013), 1512–1528.

[40] L. Minku and X. Yao. 2014. How to Make Best Use of Cross-company Data in Software Effort Estimation?. In *ICSE*. Hyderabad, 446–456.

[41] L. Minku and X. Yao. 2017. Which Models of the Past Are Relevant to the Present? A software effort estimation approach to exploiting useful past models. *ASE Journal* 24, 7 (2017), 499–542.

[42] R. Premraj, M. Shepperd, B. Kitchenham, and P. Forselius. 2005. An Empirical Analysis of Software Productivity Over Time. In *METRICS*. Como, 37.1–37.10.

[43] F. Sarro and A. Petrozziello. 2018. Linear Programming as a Baseline for Software Effort Estimation. *ACM TOSEM* 27, 3 (2018), 12.1–12.28.

[44] F. Sarro, A. Petrozziello, and M. Harman. 2016. Multi-Objective Effort Estimation. In *ICSE*. 619–630.

[45] M. Shepperd and S. McDonell. 2012. Evaluating Prediction Systems in Software Project Estimation. *IST* 54, 8 (2012), 820–827.

[46] M. Shepperd and C. Schofield. 1997. Estimating Software Project Effort Using Analogies. *IEEE TSE* 23, 12 (1997), 736–743.

[47] K. Srivasan and D. Fisher. 1995. Machine Learning Approaches to Estimating Software Development Effort. *IEEE TSE* 21, 2 (1995), 126–137.

[48] S. Tong, Q. He, Y. Chen, Y. Yang, and B. Shen. 2016. Heterogeneous Cross-Company Effort Estimation through Transfer Learning. In *APSEC*. Hamilton.

[49] B. Turhan and E. Mendes. 2014. A Comparison of Cross- versus Single- Company Effort Prediction Models for Web Projects. In *Euromicro Conference on Software Engineering and Advanced Applications*. Verona, Italy, 285–292.

[50] A. Vargha and H. D. Delaney. 2000. A critique and improvement of the CL common language effect size statistics of McGraw and Wong. In *Journal of Educational and Behavioral Statistics*, Vol. 25. 101–132.

[51] P.A. Whigham, C. A. Owen, and S. G. MacDonell. 2015. A Baseline Model for Software Effort Estimation. *ACM TOSEM* 24, 3 (2015), 20.1–20.11.

[52] I. Wieczorek and M. Ruhe. 2002. How Valuable Is Company-Specific Data Compared to Multi-Company Data for Software Cost Estimation?. In *IEEE METRICS*. Ottawa, 237–246.