How to Make Best Use of Cross-Company Data in Software Effort Estimation?

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

ABSTRACT

Previous works using Cross-Company (CC) data for making Within-Company (WC) Software Effort Estimation (SEE) try to use CC data or models directly to provide predictions in the WC context. So, these data or models are only helpful when they match the WC context well. When they do not, a fair amount of WC training data, which are usually expensive to acquire, are still necessary to achieve good performance. We investigate how to make best use of CC data, so that we can reduce the amount of WC data while maintaining or improving performance in comparison to WC SEE models. This is done by proposing a new framework to learn the relationship between CC and WC projects explicitly, allowing CC models to be mapped to the WC context. Such mapped models can be useful even when the CC models themselves do not match the WC context directly. Our study shows that a new approach instantiating this framework is able not only to use substantially less WC data than a corresponding WC model, but also to achieve similar/better performance. This approach can also be used to provide insight into the behaviour of a company in comparison to others.

Categories and Subject Descriptors

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

General Terms

Experimentation, Algorithms, Management

Keywords

Software effort estimation, cross-company learning, transfer learning, online learning, ensembles of learning machines

ICSE'14, May 31 – June 7, 2014, Hyderabad, India
ACM 978-1-4503-2756-5/14/05
http://dx.doi.org/10.1145/2568225.2568228

1. INTRODUCTION

Software Effort Estimation (SEE) is the process of estimating the effort required to develop a software project. It is a task of strategic importance in project management, as software effort is the major contributing factor for software cost. Due to the importance of this task, many automated methods for software cost or effort estimation have been proposed, including several machine learning approaches [28].

Automated methods for creating SEE models rely on a set of training examples/data which can be used for training (building) the models. The training examples usually must contain information on completed projects in the form of input features and target. Input features can be, for example the project's functional size, development type, language type, etc. The target is the true required effort of the project. However, collecting Within-Company (WC) training examples, i.e., information on projects completed by the company for which we are interested in providing predictions, takes time. Moreover, even though collecting certain input features can be relatively cheap and automated by using some search mechanism in the project database, collecting the true effort can be very expensive and involve a lot of human effort. As a result, one of the major challenges in SEE is that WC training sets are typically small, and building SEE models based solely on such small training sets can result in poor performance.

In order to overcome this problem, several studies have attempted to use Cross-Company (CC) training examples from different companies to augment existing WC training sets or to avoid their need [9, 20, 17]. We will use the term CC to refer to other companies than the one for which we are interested in providing estimations. Lately, approaches that use CC training examples have also been referred to as transfer learning approaches [11]. All existing SEE CC approaches so far try to use CC training examples or models *directly* to provide predictions in the WC context. As a result, they are only useful when they reflect the WC context well, i.e, when the effort required by another company to develop a certain software project would be the same as the effort required by the company that we are interested in. When this is not the case, which can happen in practice [20], these training examples or models perform poorly in the WC context, and a fair amount of WC training examples would still be necessary to achieve good performance.

No SEE approach so far has tried to *map* CC training examples or models to the WC context, i.e., to learn a function that transforms the effort of CC training examples or models to the effort that would be required in the company

that we are interested in. If such mapping is possible, then CC training examples or models that would not be helpful when used directly in the WC context could become helpful after being mapped, and less WC training examples would be necessary for building the SEE model. With that in mind, this paper aims at answering the following question: **If we map CC models to the WC context, can we reduce the amount of WC training examples used in the learning while maintaining or improving the performance obtained by a WC model trained on more WC examples?**

In order to answer this question, we define a CC learning scenario that considers the relationship between CC and WC training examples explicitly. After that, we propose the first framework for learning this relationship and mapping CC models to the WC context, i.e., a framework designed to make best use of CC data for SEE. This framework is then illustrated by a new approach called Dynamic Cross-company Mapped Model Learning (Dycom). This approach is evaluated on five databases and is able to substantially reduce the amount of WC projects necessary for the learning while obtaining similar or better performance than a WC model trained on more WC examples. This demonstrates the success of our proposed framework and the importance of considering the relationship between CC and WC training examples explicitly.

Dycom is designed and evaluated considering an online learning scenario where additional WC training examples may become available with time and the chronology of the projects is important. The reason why it is important to consider chronology is that the company may suffer changes with time that can affect the performance of SEE models [14, 20], i.e., SEE operates in a *dynamic* as opposed to a *static* environment. So, even though CC models may be mapped to the WC context using a certain function at a certain moment in time, another function may be necessary to make the mapping if/when the WC context changes.

2. RELATED WORK

Several SEE works have tried to use CC training examples or models directly in the WC context as an attempt to deal with the problem of small WC training sets. Kitchenham et al. [9] performed a systematic literature review with the main objective of investigating under what circumstances individual organizations would be able to rely on CC SEE models. Even though some trends were observed, no conclusions could be drawn in terms of under what circumstances CC data would be useful or not. McDonell and Shepperd [16] also performed a systematic literature review to investigate what evidence is there that CC SEE models are at least as good as WC estimation models. They point to a lack of strong evidence to support either CC or WC models. Both systematic reviews mention that the level of heterogeneity of the single-company being estimated may vary and influence WC models' performance. This has influenced more recent works that use local learning approaches to deal with the heterogeneity of WC and CC data sets [22, 17].

The works retrieved by the systematic reviews [9, 16] found WC models to be either statistically similar or worse than CC models. Some of the studies used CC training examples to augment their existing WC training sets. The resulting WC+CC training set was then used to build a model to make predictions in the WC context. For example, Lefley and Shepperd [13] tested this strategy using several learning machines based on the Finish database. In particular, they ensured that the training examples were projects completed up to a certain date, and that the test examples were projects yet to start by this date. Very few other studies considered chronological splitting [15]. Some other studies recalibrated stepwise models obtained from WC+CC data by using only CC training examples [7, 8]. So, the CC model is not independent of the WC data. For example, Kitchenham and Mendes [8] performed a study on web effort estimation based on the Tukutuku database using this approach. Other studies used solely CC training examples to build the CC predictive models [2, 29]. For example, Briand et al. [2] performed a study using the European Space Agency software project database using several different learning machines to create CC and WC models. They compared the performances resulting from cross-validation using only the WC data set to the performances of CC models trained only on a CC data set but tested on the WC data set.

Most of these works assumed the SEE environment to be static and treated CC training examples as if they were from the same context of the WC training examples. It is reasonable that, when the context of the CC and WC examples is different, CC models would perform poorly in the WC context. When the contexts are similar, then CC models may perform similarly to WC models. More recent works tried to eliminate projects from predictions when they were likely to result in poor estimations. For example, Kocaguneli et al. [12] used a filtering mechanism based on binary trees. This approach obtained very encouraging results, as models trained solely with different types of projects (or projects from different centres of a company) from the ones we would like to predict obtained overall similar performance to models trained on projects of the same type (or from the same centre). However, this approach assumes a static SEE environment and does not consider that different companies may behave considerably differently from each other. In addition, even though it has been evaluated in the context of different centres of a company, it is unclear how well it would behave in a strict CC vs WC scenario.

An insightful work [17] in the context of SEE and Software Defect Prediction (SDP) clustered WC+CC examples, and then created prediction rules for each cluster. 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 recommend to cluster WC+CC examples, but to learn rules using solely the CC examples from the envied cluster. However, even though the paper is written for both SEE and SDP, this particular conclusion was drawn based solely on SDP data. The WC models are also likely to have been trained on very small subsets of the WC data, possibly hindering their performance. This approach uses the CC examples directly in the WC context and does not consider the dynamic aspects of the SEE task.

Another work [20] compares the performance of local models (regression trees) trained only on WC and only on CC data throughout time. It confirms that different companies can have different contexts and that a certain company may suffer changes over time. These changes can cause it to

behave more/less similarly to other companies, making CC data beneficial or detrimental depending on the moment in time. An approach called Dynamic Cross-company Learning (DCL) was designed to identify when CC data is beneficial and then use it to improve performance in comparison to WC models. DCL was the first SEE approach able to use CC models to achieve better performance in comparison to WC models, showing that it is helpful to check how relevant CC models are to the current WC context before using them for WC predictions. However, as the CC models are only useful when they match the current WC context reasonably well, DCL still requires a fair amount of WC examples to achieve good performance during the periods when the CC models are not useful.

As all existing CC works attempt to use CC training examples or models directly in the WC context, there is currently no solution for the problem of small WC training sets when the CC data do not match the WC context well. As different companies are likely to behave differently, such situation is very likely to happen in practice. Our work is the first one to consider mapping CC models to the WC context so that the mapped models can be useful even when the CC models themselves do not match the WC context. A successful mapping would reduce the amount of WC training examples required for achieving good performance.

3. CC SEE LEARNING SCENARIO

Different from previous work, we consider for the first time that there is a relationship between the SEE context of a certain company and other companies. We formalise the relationship between two companies C_A and C_B as follows:

$$f_A(\mathbf{x}) = g_{BA}(f_B(\mathbf{x})) \quad (1)$$

where C_A is the company in which we are interested, C_B is another company (or a section of this other company), f_A is the true function that provides the required effort to C_A, f_B is the true function that provides the required effort to C_B, g_{BA} is a function that maps the effort from the context of C_B to the context of C_A, and $\mathbf{x} = [x_1, x_2, \cdots, x_n]$ are the input features of a software project. As an illustrative example, consider the software projects in table 1. In this case, the true effort of a project in C_A is 1.2 times the effort that would be required in C_B, i.e., $f_A(\mathbf{x}) = g_{BA}(f_B(\mathbf{x})) = 1.2 \cdot f_B(\mathbf{x})$. Even though the relationship between the effort in C_A and C_B is linear in this example, our CC learning scenario does not restrict g_{BA} to linear functions. Note also that f_A and f_B can be functions of any type. For instance, f_A (or f_B) could be composed of sub-functions representing different clusters of the company's data in order to represent the level of heterogeneity within a company [22, 17]. In practice, it is also likely that there will be some noise in the efforts.

Given equation 1, the learning task of creating an SEE model to a certain company C_A can involve the task of learning the relationship between C_A and other companies, i.e., learning how to map CC training examples or models from the context of other companies to the context of C_A.

4. CC MODEL MAPPING FRAMEWORK

Based on equation 1, we propose a framework for learning SEE models for a company C_A based on mapping CC models to C_A's context. It can be used both when WC and CC training examples arrive continuously (online) and when they comprise pre-existing sets of fixed size (offline).

4.1 Mapping One CC SEE Model

This section explains the framework to learn a function $\hat{g}_{BA}$ to map effort estimations given by an SEE model $\hat{f}_B$ from company C_B to the context of a company C_A. The learning process is illustrated in figure 1.

WC training data: Assume that training examples from C_A are made available. The number of WC examples can be very small, and a WC model created based on it may not be accurate. So, we would like to improve SEE for C_A by using CC training data.

CC training data: Assume that training examples from C_B (or from a section of C_B) are made available. The input features and target of these examples must be compatible with the WC training examples.

CC SEE model: Training examples from C_B are used for building an SEE model $\hat{f}_B$ for the context of C_B. If C_B does not wish to provide its raw training training examples to C_A, it could provide $\hat{f}_B$ directly to C_A.

Mapping function: Given the WC training examples from C_A and the CC SEE model $\hat{f}_B$, the learning task is then to learn a *mapping function* $\hat{g}_{BA}$ to map SEEs provided by $\hat{f}_B$ to the context of C_A. In order to do so, an appropriate training set needs to be derived from the WC training examples and $\hat{f}_B$. This is done as follows. For each training example $(\mathbf{x}, y)$ in the WC training set, use $\hat{f}_B$ to provide an effort estimation $\hat{f}_B(\mathbf{x})$. Then, create an example $(\hat{f}_B(\mathbf{x}), y)$ for training $\hat{g}_{BA}$. It is hoped that the task of learning $\hat{g}_{BA}$ is not more difficult than the task of learning a whole WC model $\hat{f}_A$ based solely on the WC training examples, as this would reduce the amount of WC training examples required for the learning.

Mapped SEE model: Once the mapping function is learnt, then an SEE for a project from C_A described by the input features $\mathbf{x} = [x_1, x_2, \cdots, x_n]$ would be provided by the following function, named as *mapped model*:

$$\hat{f}_A(\mathbf{x}) = \hat{g}_{BA}(\hat{f}_B(\mathbf{x})). \quad (2)$$

4.2 Using More than One CC SEE Model

If the SEE model $\hat{f}_B$ and the mapping function $\hat{g}_{BA}$ learnt using the framework explained in section 4.1 were perfect, then the mapped model would be a perfect SEE model to C_A. However, it is very unlikely that an SEE model would be a perfect, especially considering the limited amount of training examples and the difficulty of the task. In order to improve the SEEs given to C_A, CC training examples from more than one other company (or sections of a company) can (and should) be used. The framework explained in section 4.1 can be used to learn the mapping function $\hat{g}_{BiA}$ for the CC model corresponding to each company (or section of a company) C_{Bi} $(1 \leq i \leq M)$. Each of the M mapped models would provide effort estimations to C_A as shown in equation 2. So, for each WC project to be estimated, M different estimations can be provided. An approach implementing this framework would then need to decide which of these estimations should be used, and/or how these estimations should be combined into a single and more trustful estimation to be given to C_A. Combining estimations of different models has been showing to improve SEE considerably [22, 20]. In addition to the mapped models, one may also wish to combine a WC model $\hat{f}_{W_A}$ trained on C_A's training examples.

Table 1: Example of relationship between company C_A and company C_B. In this case, the true effort in person-hours for a given project in C_A is 1.2 times its true effort in person-hours in C_B.

ID	Functional Size	Development Type	Language Type	C_B's True Effort	C_A's True Effort
0	100	Enhancement	3GL	500	600
1	300	Re-development	4GL	1300	1560
2	400	New Development	4GL	2000	2400
3	500	New Development	3GL	3000	3600

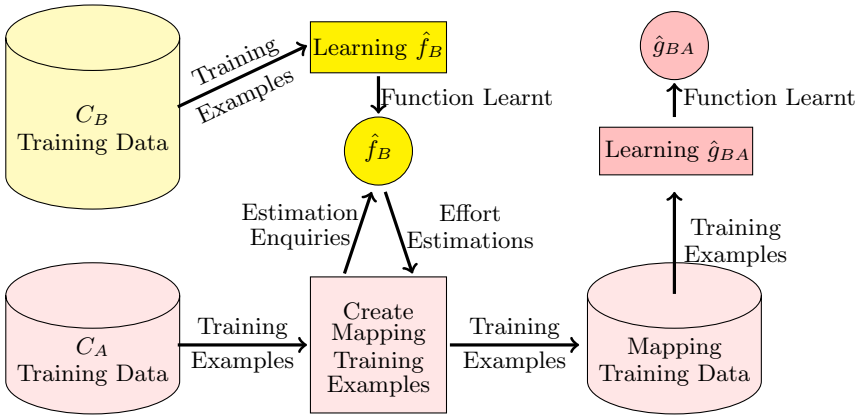


Figure 1: Framework for learning function $\hat{g}_{BA}$ to map estimations given by an SEE model $\hat{f}_B$ from company C_B into the context of a company C_A.

4.3 Using a Mixed CC Training Set

A certain company C_A may have access to a set of CC training examples from several different companies without information on which examples belong to which individual companies among the ones that provided the examples. This would be the case of a company C_A using data from the International Software Benchmarking Standards Group (ISBSG) [6], for example. Nevertheless, it would still be possible to use our framework. For instance, the CC training examples could be split into different sets according to their corresponding productivities, as done in section 7, or according to some clustering technique based on their input features and targets. In this way, similar CC examples would be grouped together into different C_{Bi}, $(1 \leq i \leq M)$ sets, where each C_{Bi} is a "virtual" (imaginary) company that develops projects with the characteristics of the projects grouped together. Each virtual company would represent a mix of the behaviours of the different companies that provided the CC projects. If appropriate functions $\hat{f}_{C_{Bi}}$ are learnt, the context of these virtual companies C_{Bi} can be mapped to the context of C_A, turning them into useful information to C_A.

5. ONLINE LEARNING SCENARIOS

As explained in section 4, the CC model mapping framework can be used both for online and offline learning. In this paper, we propose an instantiation of the framework where new WC projects arrive with time and CC training examples comprise pre-defined sets, similarly to [20]. Such situation could occur, for example, when a company acquires sets of CC training examples without a contract to receive updates of these sets. Another situation would be when a company adopts a CC training set available in the web, whose size is not increasing with time.

As companies are evolving entities, changes can happen that would affect the effort required for them to develop a software project [20, 14]. Examples of changes are training being provided to employees, new employees being hired, new programming languages being introduced, the company starting to take new types of projects, etc. So, it is important to consider the chronology (e.g., the date of completion) of the projects of the company for which we are interested in providing estimations. Additionally, if this company suffers changes that affect the effort that it would require to develop a software project, the model used for providing such estimations should be able to adapt to such changes.

We formulate SEE as an online learning problem in which a new WC project is completed at each *time step* (point in time). Whenever a new WC project is completed, we wish to provide up-to-date estimations of the effort for the next ten WC projects to be (or being) developed, based on the latest available SEE model. The number of WC projects for which up-to-date estimations should be provided at each time step depends on the needs of the company being estimated. Investigation of values different from ten is left as future work. As an important aim of the current work is to analyse whether it is possible to reduce the amount of WC training examples required for building an SEE model, we consider two different scenarios:

1. Each completed WC project contains both information on its input features and on its true required effort.

2. The WC projects that arrive at every p $(p > 1)$ time steps contain both information on their input features and true required effort. All remaining WC projects contain only the information on the input features, whereas the true required effort is missing. So, even though an effort estimation is required for all WC projects, only a few of them are used as training examples once they are completed.

6. DYCOM

In this section, we present Dynamic Cross-company Mapped Model Learning (Dycom), an instantiation of the framework presented in section 4. Dycom can be used with CC

training examples from several different companies, as described in sections 4.2 and 4.3, and operates under the online learning scenario 2 explained in section 5.

WC training data: Similarly to section 4, we will refer to the company for which we are interested in providing predictions as C_A. WC training examples from C_A are incoming as described in section 5.

WC SEE model: Our framework allows a WC model to be trained if desired. In Dycom, whenever a new WC training example arrives, it is used to train a model $\hat{f}_{W_A}$.

CC training data: The CC training examples are available beforehand as explained in section 5. They are split into sections based on some clustering algorithm, or on their productivity, or on the size of the projects. Each section C_{Bi} is considered as a separate CC training set. For example, if there are N companies and the training examples from each company are split into S sections, then there will be $M = N \cdot S$ different CC training sets. If it is not known from what company each CC training example comes, then the whole CC training set is split into different sections, as if $N = 1$. The reason for the splitting will be explained in the paragraph on mapping functions.

CC SEE models: Each of the M CC training sets is used to create a different CC model $\hat{f}_{Bi}$ $(1 \leq i \leq M)$.

Mapping functions: Whenever a new WC training example arrives, each model $\hat{f}_{Bi}$ is asked to perform an SEE, each SEE is then used to create a mapping training example $(\hat{f}_{Bi}(\mathbf{x}), y)$, and the corresponding $\hat{g}_{BiA}$ is trained with it, as described in section 4.1. Dycom assumes that the relationship formalised in equation 1 can be modelled reasonably well by linear functions of the format $\hat{g}_{BiA}(\hat{f}_{Bi}(\mathbf{x})) = \hat{f}_{Bi}(\mathbf{x}) \cdot b_i$ when different sections containing relatively more similar CC training examples are considered separately. This is the reason to split CC training examples into different sections as explained previously. Learning this function means learning the factor b_i, which is done using equation 3:

$$b_i = \begin{cases} 1, & \text{if no mapping training example has been received yet;} \\ \dfrac{y}{\hat{f}_{Bi}(\mathbf{x})}, & \text{if } (\hat{f}_{Bi}(\mathbf{x}), y) \text{ is the first mapping training example;} \\ lr \cdot \dfrac{y}{\hat{f}_{Bi}(\mathbf{x})} + (1 - lr) \cdot b_i, & \text{otherwise.} \end{cases} \quad (3)$$

where $(\hat{f}_{Bi}(\mathbf{x}), y)$ is the mapping training example being learnt, lr $(0 < lr < 1)$ is a pre-defined smoothing factor and the factor b_i in the right side of the equation represents the latest value of b_i before receiving the current mapping training example.

The mapping function performs a direct mapping $b_i = 1$ while no mapping training example is received. When the first mapping training example is received, b_i is set to the value $y/\hat{f}_{Bi}(\mathbf{x})$. This gives a perfect mapping for the example being learnt, as $\hat{f}_{Bi}(\mathbf{x}) \cdot b_i = \hat{f}_{Bi}(\mathbf{x}) \cdot y/\hat{f}_{Bi}(\mathbf{x}) = y$. For all other mapping training examples received, exponential smoothing with smoothing factor lr is used to set b_i. This is the simple weighted average of the value that would provide a perfect mapping for the current mapping example and the previous value of b_i, which was calculated based on the previous mapping examples. Higher smoothing factor lr will cause more emphasis on the most recent mapping training examples and higher adaptability to changing environments, whereas lower lr will lead to a more stable mapping function. So, the smoothing function allows learning mapping functions that provide good mappings based on previous mapping examples, while allowing adaptability to changes that may affect a company's required software efforts.

Mapped SEE model: As explained in section 4.2, each mapped model $\hat{g}_{BiA}(\hat{f}_{Bi})$ and the WC model $\hat{f}_{W_A}$ can provide an SEE in the WC context when required. The SEE given by Dycom is the weighted average of these $M + 1$ estimations:

$$\hat{f}_A(\mathbf{x}) = \left[\sum_{i=1}^{M} w_{Bi} \cdot \hat{g}_{BiA}(\hat{f}_{Bi}(\mathbf{x})) \right] + w_{W_A} \hat{f}_{W_A}(\mathbf{x}), \quad (4)$$

where the weights w_{Bi} and w_{W_A} represent how much we trust each of the models, are positive and sum to one. So, Dycom uses an ensemble of mapped and WC SEE models.

Weights: The weights are initialised so that they have the same value for all models being used in the ensemble and are updated in a similar way to the weights used in [20]. Whenever a new WC training example is made available, the model which provided the lowest absolute error is considered to be the *winner* and the others are the *losers*. The losers have their weights multiplied by a pre-defined parameter β $(0 < \beta \leq 1)$, and then all weights are normalised so that they sum to one.

Algorithm 1 presents Dycom's learning process. Dycom first learns the CC models (line 3). The weight associated to each CC model is initialised to $1/M$, so that each model has equal weight and all weights sum to one (line 4). The mapping functions are initialised to use $b_i = 1$ (line 5). Before any WC training example is made available, the weight w_{W_A} corresponding to the WC model is initialised to zero (line 7), because this model has not received any training yet. In this way, the CC models can be used to make predictions while there is no WC training example available. After that, for each new WC training example, the weights are updated (lines 9–19). If the WC training example is the first one, the weight of the WC model needs to be set (line 17). Then, the mapping training examples are created and used to update the corresponding mapping functions (lines 22 and 23). Finally, the WC model is updated with the WC training example (line 26).

7. DATABASES

This section presents the databases used in the evaluation of Dycom. Five different databases were used: KitchenMax, CocNasaCoc81, ISBSG2000, ISBSG2001 and ISBSG. These include both data sets derived from the PRedictOr Models In Software Engineering Software (PROMISE) Repository [18] and the International Software Benchmarking Standards Group (ISBSG) Repository [6]. Each database contains a WC data set and three CC data sets derived based on the projects' productivity.

7.1 KitchenMax

The database KitchenMax is composed of Kitchenham and Maxwell, which are two SEE data sets available from the PROMISE Repository. Kitchenham's detailed description can be found in [10]. It comprises 145 maintenance and development projects undertaken between 1994 and 1998 by a single software development company. Maxwell's detailed

Algorithm 1 Dycom

Parameters:
D_{Bi} $(1 \leq i \leq M)$: CC training sets.
β: factor for decreasing model weights

{Learn CC base models:}
for each CC training set D_{Bi} **do**
  Create CC model $\hat{b}_{Bi}$ using D_{Bi}
  $w_{Bi} = \frac{1}{M}$ {Initialise weight.}
  $b_i = 1$ {Initialise mapping function.}
end for
$w_{W_A} = 0$ {Initialise WC model weight.}
for each new WC training example $(\mathbf{x}, y)$ **do**
  {Update weights:}
  for each model $\hat{f}_{Bi}$ and $\hat{f}_{W_A}$ **do**
    Determine the model's estimation to $\mathbf{x}$.
    Calculate the absolute error AE_{Bi} (or AE_{W_A}).
  end for
  Determine loser models based on their AE.
  Multiply loser models' weights by β.
  if $(\mathbf{x}, y)$ is the first WC training example **then**
    $w_{W_A} = \frac{1}{M+1}$
  end if
  Divide each weight by the sum of all weights.
  {Update mapping functions:}
  for each model $\hat{f}_{Bi}$ **do**
    Create mapping example $(\hat{f}_{Bi}(\mathbf{x}), y)$.
    Use $(\hat{f}_{Bi}(\mathbf{x}), y)$ to update $\hat{g}_{BiA}$ based on Eq. 3.
  end for
  {Update WC model:}
  Update WC model $\hat{f}_{W_A}$ using $(\mathbf{x}, y)$.
end for

description can be found in [25]. It contains 62 projects from one of the biggest commercial banks in Finland, covering the years 1985 to 1993 and both in-house and outsourced development. In order to make these data sets compatible, a single input feature (functional size) was used. Still, Maxwell uses functional size, whereas Kitchenham uses adjusted functional size. An appropriate mapping function should be able to overcome this problem. There were no functional size feature values missing. The target is the effort in person-hours.

Kitchenham was considered as the WC data, and was sorted according to the actual start date plus the duration. This sorting corresponds to the exact completion order of the projects. Maxwell was considered as the CC data and was split into three CC sets for use with Dycom according to their productivity in terms of effort divided by functional size. The ranges used for the different CC sets are shown in table 2 and were chosen to provide similar size partitions. This process could be easily automated in practice. WC projects were not split.

7.2 CocNasaCoc81

The database CocNasaCoc81 is composed of Cocomo Nasa and Cocomo 81, which are two SEE data sets available from the PROMISE Repository. Cocomo Nasa contains 60 Nasa projects from 1980s-1990s and Cocomo 81 consists of the 63 projects analysed by Boehm to develop the software cost estimation model COCOMO [1] first published in 1981. Both data sets contain 16 input features (15 cost drivers [1] and number of lines of code) and one target (software effort in person-months). Cocomo 81 contains an additional input feature (development type) not present in Cocomo Nasa, which was thus removed. These data sets contain no missing values.

Table 2: Productivity ranges for CC data sets.

CC Data	Productivity Band	# Examples
Maxwell	High: (3.85,7.40]	17
	Medium: (7.40,15.30]	24
	Low: (15.30,38.00]	21
Cocomo 81	High: [0.7,2.85]	19
	Medium: (2.85,6.60]	20
	Low: (6.60,49.00]	24
ISBSG2000	High: [0.00,5.00]	56
	Medium: (5.00,13.00]	57
	Low: (13.0,155.70]	55
ISBSG2001	High: [0.00,6.00]	72
	Medium: (6.00,14.00]	79
	Low: (14.00,155.70]	73
ISBSG	High: [0.00,10.00]	291
	Medium: (10.00,20.00]	250
	Low: (20.00,424.90]	285

Cocomo Nasa's projects were considered as the WC data and Cocomo 81's projects were considered as the CC data. Cocomo Nasa provides no information on whether the projects are sorted in chronological order. The original order of the Cocomo Nasa projects was preserved in order to simulate the WC projects' chronology. Even though this may not be the true chronological order, it is still useful to evaluate whether approaches are able to make use of mapped CC models when/if they are beneficial. The three CC data sets for Dycom were created by separating Cocomo 81's projects based on the productivity in terms of effort divided by the number of lines of code using the ranges in table 2. This database has also been used in [20], where it was shown that Cocomo 81's projects can sometimes be useful and sometimes detrimental in predicting Cocomo Nasa's projects.

7.3 ISBSG Databases

Three SEE databases were derived from ISBSG Release 10, which contains software project information from several companies. Information on which projects belong to a single company for composing a WC data set have been provided to us upon request. The databases are:

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

These databases have been previously used in [20]. Information on how they were preprocessed can be found in that paper and is not included here due to space limitations. Four input features (development type, language type, development platform and functional size) and one target (software effort in person-hours) were used. The WC projects

were sorted based on the implementation date to compose a stream of incoming projects. Dycom further uses a separation of CC projects into different training sets. This was done by splitting projects according to their normalised level 1 productivity rate in hours per functional size unit, provided by the repository. The ranges used for creating the CC sets are shown in table 2. It was shown in [20] that regression trees (local models [22]) trained on each CC set were better, similar or worse than regression trees trained on the WC projects depending on the moment in time.

8. EVALUATION OF DYCOM

This section presents the experiments performed with the aim of determining whether mapping CC models to the WC context allows us to reduce the amount of WC training examples used in the learning while maintaining or improving performance in comparison to a WC model trained with more WC examples. These experiments also work as an evaluation of Dycom, as is used as the mapping learning approach.

8.1 Experimental Setup

Dycom can be used with any base learner. In this work, Regression Trees (RTs) were used as the base learners in the experiments. RTs were chosen because they are local approaches in which estimations are based on the projects that are most similar to the project being predicted. This can help dealing with the heterogeneity within each data set [22]. RTs have been shown to achieve good performance for SEE in comparison to several other approaches [22]. We used the RT implementation *REPTree* provided by WEKA [5], where splits are created so as to minimise the variance of the targets of the training examples in the nodes. Two different approaches were compared: RT and Dycom-RT.

RT: RTs were created to reflect WC online learning. Whenever a new WC training example was provided, the current RT was discarded and a new RT was trained on all projects so far (including the one received at the current time step). This RT was then used to predict the next ten WC projects. This approach considers that all WC completed projects contained known true effort, i.e., every time step received a WC training example (online scenario 1).

Dycom-RT: Dycom was used with RTs as the CC and WC models under the online scenario 2 with $p = 10$, i.e., a new WC training example was provided only at every 10 time steps. So, Dycom-RT uses only 10% of the WC training examples used by the RT explained in the paragraph above. The WC RT used by Dycom was rebuilt from scratch using all WC training examples so far whenever a new WC example was made available.

Both Dycom and the WC approach used the same base learner in the experiments, ensuring that the comparison is fair. The study of Dycom with other base learning approaches is left as future work.

At each time step, the performance on the next ten examples was evaluated based on several different measures: Mean Absolute Error (MAE), Standardised Accuracy (SA), Root Mean Squared Error (RMSE), correlation coefficient (Corr), Logarithmic Standard Deviation (LSD), Mean Magnitude of the Relative Error (MMRE) and percentage of predictions within 25% of the actual value (PRED(25)). The equations to calculate these measures are the following, where T is the number of examples used for evaluating the performance, y_i is the actual effort for the example i, and $\hat{y}_i$ is the estimated effort for example i:

- $MAE = \frac{1}{T}\sum_{i=1}^{T}|\hat{y}_i - y_i|$;
- $SA = \left(1 - \frac{MAE}{MAE_{rguess}}\right) \cdot 100$, where MAE is the MAE of the approach being evaluated and MAE_{rguess} is the MAE of 1000 runs of random guess. Random guess is defined here as uniformly randomly sampling the true effort over all the WC projects received up to the current time step. It is calculated based on the online scenario 1;
- $RMSE = \sqrt{\frac{\sum_{i=1}^{T}(\hat{y}_i - y_i)^2}{T}}$;
- $Corr = \frac{\sum_{i=1}^{T}(\hat{y}_i - \bar{\hat{y}})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{T}(\hat{y}_i - \bar{\hat{y}})^2}\sqrt{\sum_{i=1}^{T}(y_i - \bar{y})^2}}$, where $\bar{\hat{y}}$ and $\bar{y}$ are the average predicted and average actual efforts, respectively;
- $LSD = \sqrt{\frac{\sum_{i=1}^{T}\left(e_i + \frac{s^2}{2}\right)^2}{T-1}}$, where s^2 is an estimator of the variance of the residual e_i and $e_i = \ln y_i - \ln \hat{y}_i$;
- $MMRE = \frac{1}{T}\sum_{i=1}^{T} MRE_i$, where $MRE_i = |\hat{y}_i - y_i|/y_i$;
- $PRED(25) = \frac{1}{T}\sum_{i=1}^{T} \begin{cases} 1, & \text{if } MRE_i \leq \frac{25}{100} \\ 0, & \text{otherwise.} \end{cases}$.

These measures have been chosen because they emphasise a variety of different behaviours, evaluating SEE from different angles. MAE has been recommended by [26] for being unbiased towards under or overestimations. SA is an unbiased measure that allows for interpretability – it is viewed as the ratio of how much better an approach is than random guess [26]. So, it can also be used to give a better idea of the magnitude of the differences in performance. RMSE is a popular measure in the machine learning community which emphasizes large errors more. Corr is widely used in sciences as a measure of the strength of linear dependence between two variables. In the case of SEE, the two variables are the estimated and the actual effort [3]. LSD uses the residual in the log-scale, which is independent of size (i.e., homoscedastic) [4]. MMRE and PRED(25) are measures biased towards underestimations and can behave quite differently from other measures [4, 23, 21]. They are reported here to provide insight into the behaviour of the approaches when analysed together with MAE. In addition to these measures, the standard deviation (StdDev) of MAE across time steps has also been used. This is because it is desirable to have SEE models that are not only accurate, but also whose errors do not vary much for different projects. A more stable SEE model is more reliable, because it gives a better idea of the error that is likely to happen when estimating a new project.

The databases used in the experiments are the ones explained in section 7. Dycom's parameter β was set to the default value of 0.5, which has been used previously in the literature for similar weight update mechanisms [20]. Dycom's parameter lr was set to 0.1 after some preliminary investigation with 0.1 and 0.05. The parameters used with each RT were the ones more likely to obtain good results in previous work [22]: minimum total weight of 1 for the instances in a leaf, and minimum proportion of the variance on all the data that need to be present at a node in order for splitting to be performed 0.0001. A single execution was performed for each data set, as we used deterministic RTs.

Table 3: Overall Average Performance Across Time Steps.

Database	Approach	MAE	StdDev	SA	RMSE	Corr	LSD	MMRE	PRED(25)
KitchenMax	RT	2441.0241	2838.2375	30.1782	4850.3387	0.4350	1.2221	102.2562	22.5185
	Dycom-RT	2208.6522	2665.4276	36.8249	4287.4476	0.6416	0.8809	114.0218	27.5556
	P-value	3.82E-11	6.35E-01	–	1.46E-12	1.62E-16	4.25E-21	3.80E-04	4.12E-05
CocNasaCoc81	RT	319.4572	250.2325	33.1366	477.2357	0.6427	0.8623	188.8098	24.4000
	Dycom-RT	161.7917	105.7591	66.1365	243.6504	0.8885	0.6671	91.2384	28.6000
	P-value	4.04E-06	1.40E-11	–	5.95E-08	4.12E-07	8.82E-04	2.37E-01	2.78E-02
ISBSG2000	RT	2753.3726	1257.4586	37.0471	4133.1006	0.3554	1.4592	162.7492	23.7615
	Dycom-RT	2494.6639	1249.8400	42.9622	3741.8009	0.4515	1.1589	135.5271	19.0826
	P-value	4.72E-02	1.01E-01	–	1.83E-01	8.73E-02	1.27E-06	1.90E-01	9.60E-03
ISBSG2001	RT	3621.9598	1367.9603	11.9270	5149.6267	0.1658	1.8110	395.1080	23.7288
	Dycom-RT	2543.9495	1165.8591	38.1403	3581.6573	0.5691	1.2447	228.0416	21.0169
	P-value	3.21E-06	4.16E-01	–	7.88E-06	2.29E-10	6.24E-08	1.76E-07	2.26E-01
ISBSG	RT	3253.9349	2476.0512	46.2891	4872.9193	0.4412	1.3475	160.7898	21.8966
	Dycom-RT	3122.6603	2227.9812	48.4560	4473.6527	0.5817	1.0378	162.1517	19.0805
	P-value	5.56E-02	3.54E-01	–	4.18E-02	1.90E-09	2.99E-12	2.99E-05	4.59E-02

Cells in lime (light grey) represent better values. P-values of Wilcoxon sign rank tests to compare Dycom-RT against RT for each database are also shown. For StdDev, the p-values correspond to Levene tests for equality of variances. Cells in orange (dark grey) indicate statistically significant difference when using Holm-Bonferroni corrections at the overall level of significance of 0.05 considering the five databases. P-values for SA are not shown because this measure is an interpretable equivalent of MAE.

8.2 Results

8.2.1 *Overall Average Performance*

We first analyse the overall average performances across time steps (table 3). In order to compare the overall performances between RT and Dycom-RT, we performed a Wilcoxon sign rank test for each database. For StdDev, we performed Levene test for equality of variances. For each performance measure, we used Holm-Bonferroni corrections considering five comparisons at the overall level of significance of 0.05. No tests have been done for SA, because it is an interpretable equivalent to MAE.

In terms of all performance measures but MMRE and PRED(25), Dycom-RT obtained always similar or better overall performance than RT. It is worth noting that even if Dycom-RT had obtained similar (and never better) overall performance than RT, this would still have represented a strong advantage of Dycom-RT. The reason is that Dycom-RT required much less WC training examples, saving the cost of collecting the required effort for WC projects. Our experiments show that Dycom-RT not only managed to use ten times less WC training examples, but also was able to provide similar or better overall performance for all databases. In terms of MAE (and SA), RMSE and Correlation, sometimes the differences were significant and sometimes not. The measure SA allows us to have a better idea of the magnitude of the differences in performance, as it is interpreted as how much better an approach does than random guess. For CocNasaCoc81 and ISBSG2001, the difference in SA was considerably large, being likely to have large impact in practice. In terms of LSD, there was always statistically significant difference and Dycom-RT was better for all databases. In terms of StdDev, statistically significant difference was found only for CocNasaCoc81. So, Dycom-RT and RT behaved mostly similarly in terms of stability of MAE across time steps.

In terms of MMRE and PRED(25), Dycom-RT was sometimes better, similar and worse than RT. As explained in section 8.1, these measures are biased towards underestimations. So, combined with the results in terms of MAE, these results do not imply that Dycom-RT obtained behaviour sometimes better, similar or worse than RTs, but that Dycoms sometimes managed to obtain more improvements in terms of reducing overestimations, sometimes balanced improvements, and sometimes more improvements in terms of reducing underestimations. Sometimes these improvements pushed the relative errors under the boundary of 25% of the actual effort and sometimes not.

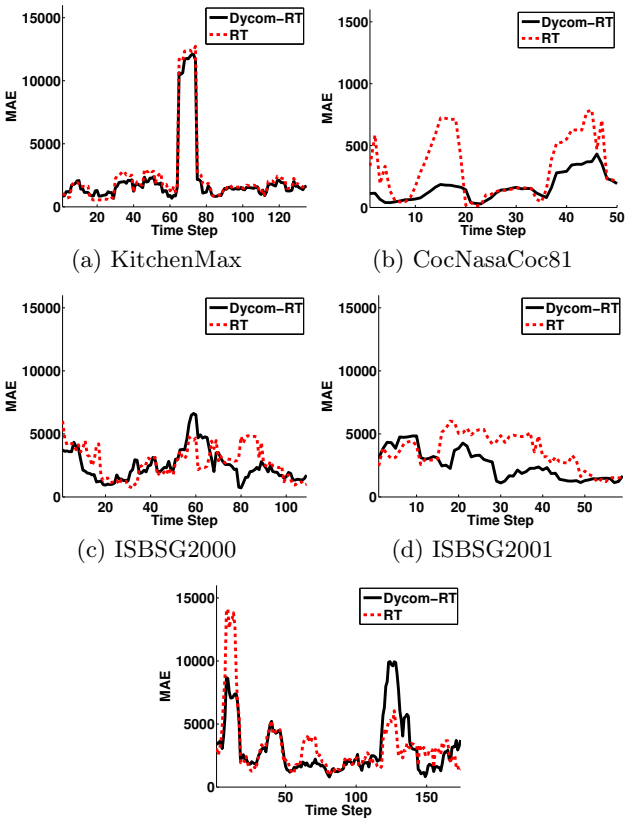
8.2.2 *Performance Throughout Time*

In addition to the overall performance across time steps, when working with online learning, it is also important to verify the performance at each time step. This allows checking whether a certain approach is better at some time steps, but worse at others. We make this analysis based on MAE, which is an unbiased measure recommended in [26]. Figure 2 shows the MAE over time for all databases.

For the databases where Dycom-RT obtained statistically better overall MAE across time steps than RT (KitchenMax, CocNasaCoc81 and ISBSG2001), there were very few time steps when Dycom-RT performed worse than RT. However, for the databases where Dycom-RT and RT obtained no statistically significant difference in terms of overall MAE (ISBSG2000 and ISBSG), Dycom-RT performed considerably better during some prolonged periods of time and considerably worse during some others. The reason for the worse performance during these periods of time is likely to be that the single company suffered some change that was not captured for a while due to the lack of WC training examples. The best interval p for the collection of true required effort of WC projects is likely to depend on how often the company suffers considerable changes. Companies that present changes more often would need smaller intervals. The interval p may also affect the ability of Dycom to deal with outlier training examples. A very interesting area of future research is the investigation of manual or algorithmic approaches to decide when projects should have their required efforts collected.

9. INSIGHT PROVIDED BY DYCOM

The mapping functions learnt by Dycom explain the relationship of the effort required by a certain company C_A in comparison to sections C_{Bi} of other companies throughout

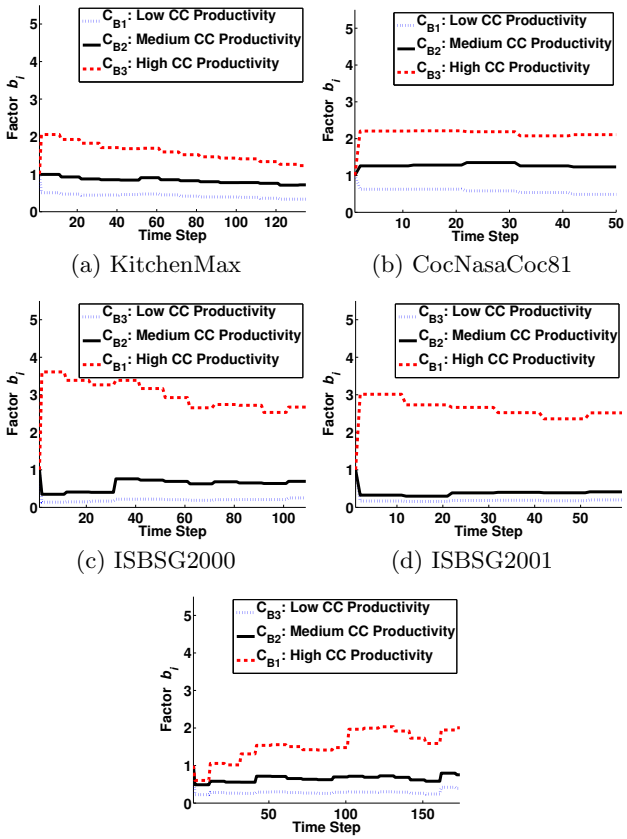


(a) KitchenMax (b) CocNasaCoc81

(c) ISBSG2000 (d) ISBSG2001

(e) ISBSG

Figure 2: MAE Throughout Time.



(a) KitchenMax (b) CocNasaCoc81

(c) ISBSG2000 (d) ISBSG2001

(e) ISBSG

Figure 3: Factor b_i Associated to Each CC Data Set Throughout Time.

time. So, it provides useful insight into the behaviour of a given company in comparison to other companies. This relationship can be visualised via plots of the factor b_i to show the need for strategic decision making towards the improvement of productivity, as well as to monitor the success of strategies adopted for such improvement. The mapping function considers the effort that different sections from other companies would require for each given project from C_A considering their input features. So, plots the factor b_i consider more information than general plots of the effort or productivity of a company throughout time, which would provide only trends without considering the individual features of the projects being developed.

Figure 3 presents the factor b_i learnt for each mapping function over time. In our experiments, the CC projects were split into three data sets according to their productivity. So, the CC sections C_{Bi} $(1 \leq i \leq 3)$ refer to a CC data set with high productivity, medium productivity and low productivity. For all databases, C_A needs more effort than the high productivity CC sections $(b_i > 1)$ and less effort than the low productivity CC sections $(b_i < 1)$. However, each single company C_A being analysed has a different behaviour with respect to their corresponding CC models, especially with respect to the high productivity ones.

For instance, KitchenMax's C_A initially behaves quite similarly to the medium productivity CC model, and then its behaviour gradually changes to become more and more similar to the high productivity CC model. Initially, a project in C_A would require around twice the effort that this same project would require by the high productivity CC model. In the end of the period observed, this figure improves to 1.2 times the effort. This demonstrates the success of the company C_A in improving its behaviour.

Another example is CocNasaCoc81, whose relationship with respect to all CC models remains stable. C_A requires around 1.25 times the effort that would be required by the medium productivity CC model. So, CocNasaCoc81's C_A may wish to adopt some strategy to become more productive. In order to decide on a strategy, one would have to ask why C_A requires more effort than the medium productivity CC model, or why C_A requires more effort than the high productivity CC model. In order to answer these questions, if C_A has access to the training examples from these CC sections, it could analyse them to find out possible reasons for C_A's lower productivity. For example, C_A could analyse the medium productivity CC projects that are most similar (e.g., nearest neighbours) to each given project that C_A wishes to improve. Or it could analyse all training examples from this CC section if C_A wishes to improve its overall behaviour. If the features of the CC projects from a given section are completely different from the WC projects, then it may be more difficult to interpret these differences and decide on a strategy to improve C_A. So, it may be a better strategy to analyse the examples from the CC set with the most similar productivity first, as these are more likely to be more similar and incur less dramatic changes for a start.

For example, table 4 shows that both the medium productivity CC section and C_A frequently used staff with high language experience. However, the medium productivity CC section used staff highly experienced in virtual machine more

Table 4: Number of projects with each feature value for the 20 CC projects from the medium productivity CC section and the first 20 WC projects.

Feature / Value	Lexp		Vexp	
	CC	WC	CC	WC
Very low	1	0	1	0
Low	1	0	4	4
Nominal	8	8	8	16
High	10	12	7	0
Very high	0	0	0	0
Extremely high	0	0	0	0

'Lexp' refers to the feature language experience and 'Vexp' refers to the feature virtual machine experience.

often than C_A. It may happen that the different companies did not judge whether staff had nominal or high experience in exactly the same way. However, the fact that the medium productivity CC section contains more projects with high virtual machine experience indicates that this feature, rather than language experience, is more likely to be one of the reasons for C_A's lower productivity. After such analysis, C_A could then adopt a certain strategy based on the differences between its project features and the CC project features to improve its productivity. For instance, it could hire staff with more virtual machine experience. The plots of b_i could then be used to analyse how successful the strategies adopted are being with time.

One might find intriguing that the same C_A provided by the ISBSG Repository behave similarly for ISBSG2000 and ISBSG2001, but differently for ISBSG. However, this behaviour is understandable, because the CC training examples used for ISBSG were not restricted to past projects. They can represent, for example, companies that are already used to certain technologies that are new to C_A. When such new technologies start being used, C_A starts requiring more effort than these companies.

10. THREATS TO VALIDITY

Internal validity regards to establishing that a certain observable event was responsible for a change in behaviour. It is related to the question "Is there something other than the treatment that could cause the difference in behaviour?" [24]. When using machine learning approaches, it is important that the approaches being compared use fair parameter choices in comparison to each other [19, 27]. In this paper, both the RTs used as WC learners and within Dycom used the same parameters, which were the ones more likely to obtain good results in the literature [22]. Dycom contains two extra parameters which were set to the same value for all databases used in this study, i.e., they were not fine tuned for each database. So, the results obtained in this paper do not depend on the user fine tuning Dycom. It is natural that a software manager with no specialist knowledge on machine learning would run these approaches with default parameters or parameters that have previously obtained good results in the literature. So, it is reasonable to perform the analysis in this way in the current paper. Future work should investigate whether Dycom's results could be improved further by fine tuning parameters.

Construct validity regards to accurately naming our measures and manipulations [24]. We used several different performance measures (MAE, StdDev, SA, RMSE, Corr, LSD, MMRE and PRED(25)). Wilcoxon statistical tests with Holm-Bonferroni corrections were used to check the statistical significance of the differences in overall performance, and SA was also used to give a better idea of the magnitude of the differences in performance and the impact that they are likely to have in practice.

External validity regards to generalizing the study's results outside the study to other situations [24]. Besides never using a WC project for training before using it for testing, we considered five databases to handle external validity. Four databases with known WC chronological order were used for evaluating Dycom. Even though the WC chronological order is not known for the other database, it can still be used to evaluate whether Dycom is able to successfully make use of CC models, contributing to the generalisation of our results. Obtaining additional databases for evaluating Dycom is difficult due to our need for non-proprietary data sets with information on which projects belong to a single-company among the projects of a cross-company data set. However, the data sets used in this study can be made available through PROMISE and ISBSG. So, researchers and companies willing to use Dycom could use the same CC data sets used in this study. Further investigation should be done into what would happen with Dycom when using other base learners than RTs, and when using other input features for the data sets.

11. CONCLUSIONS

We have introduced a new CC SEE learning scenario that considers the relationship between the required effort of WC and CC projects. We propose the first framework for learning this relationship and mapping CC models to the WC context. It is designed to make best use of CC data in SEE. The benefit of such a framework is demonstrated by a new approach Dycom, which was evaluated on five databases and was able to substantially reduce the amount of required WC training examples while maintaining or improving overall average performance in comparison to a WC model trained on more WC examples. The research question raised in section 1 was: if we map CC models to the WC context, can we reduce the amount of WC labelled projects used in the learning and still obtain similar or better performance than a WC model? The study performed in this paper gives the answer "yes" to this question. Dycom is an example of approach that successfully achieves that. Dycom can also be used to give insight into monitoring and improving the productivity of a company.

Future work includes an investigation of Dycom's sensitivity to parameter values, base learners, input features and techniques for splitting CC projects into different sections. Our proposed framework could also be used to design other approaches. For instance, approaches considering different productivity sections of WC data could also be considered. The online learning scenarios should also be studied when predicting different numbers of WC projects than ten at each time step, and techniques to decide which projects to have their effort collected should be investigated. In addition, the best type of function to describe the relationship between WC's and CC's effort should be studied.

12. ACKNOWLEDGEMENTS

The authors are most grateful to the reviewers for their constructive comments. This work was supported by EPSRC grant no. EP/J017515/1 and European Commission FP7 grant no. 270428. Xin Yao was supported by a Royal Society Wolfson Research Merit Award.

13. REFERENCES

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

[2] L. Briand, T. Langley, and I. Wieczorek. A replicated assessment of common software cost estimation techniques. In *International Conference on Software Engineering (ICSE)*, pages 377–386, Limerick, Ireland, 2000.

[3] K. Dejaeger, W. Verbeke, D. Martens, and B. Baesens. Data mining techniques for software effort estimation: A comparative study. *IEEE Transactions on Software Engineering (TSE)*, 38(2):375–397, 2012.

[4] T. Foss, E. Stensrud, B. Kitchenham, and I. Myrtveit. A simulation study of the model evaluation criterion MMRE. *IEEE Transactions on Software Engineering (TSE)*, 29(11):985–995, 2003.

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

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

[7] R. Jeffery, M. Ruhe, and I. Wieczorek. A comparative study of two software development cost modeling techniques using multi-organizational and company-specific data. *Information and Software Technology (IST)*, 42(14):1009–1016, 2010.

[8] B. Kitchenham and E. Mendes. A comparison of cross-company and single-company effort estimation models for web applications. In *Empirical Assessment in Software Engineering (EASE)*, pages 47–55, Edinburgh, 2004.

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

[10] B. Kitchenham, S. L. Pfleeger, B. McColl, and S. Eagan. An empirical study of maintenance and development estimation accuracy. *Journal of Systems and Software (JSS)*, 64:57–77, 2002.

[11] E. Kocaguneli, B. Cukic, and H. Lu. Predicting more from less: Synergies of learning. In *International NSF Sponsored Workshop on Realising Artificial Intelligence Synergies in Software Engineering (RAISE)*, pages 42–48, San Francisco, 2013.

[12] E. Kocaguneli, G. Gay, T. Menzies, Y. Yang, and J. W. Keung. When to use data from other projects for effort estimation. In *IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 321–324, Antwerp, Belgium, 2010.

[13] M. Lefley and M. Shepperd. Using genetic programming to improve software effort estimation based on general data sets. In *Genetic and Evolutionary Computation Conference (GECCO)*, pages 2477–2487, Chicago, 2003.

[14] C. Lokan and E. Mendes. Applying moving windows to software effort estimation. In *International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pages 111–122, Lake Buena Vista, Florida, USA, 2009.

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

[16] S. G. McDonell and M. Shepperd. Comparing local and global software effort estimation models – reflections on a systematic review. In *International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pages 401–409, Madrid, 2007.

[17] T. Menzies, A. Butcher, D. Cok, A. Marcus, L. Layman, F. Shull, B. Turhan, and T. Zimmerman. Local vs. global lessons for defect prediction and effort estimation. *IEEE Transactions on Software Engineering (TSE)*, 39(6):822–834, 2013.

[18] T. Menzies, B. Caglayan, Z. He, E. Kocaguneli, J. Krall, F. Peters, and B. Turhan. The promise repository of empirical software engineering data. http://promisedata.googlecode.com, 2012.

[19] T. Menzies and M. Shepperd. Special issue on repeatable results in software engineering prediction. *Empirical Software Engineering (ESE)*, 17:1–17, 2012.

[20] L. Minku and X. Yao. Can cross-company data improve performance in software effort estimation? In *International Conference on Predictive Models in Software Engineering (PROMISE)*, pages 69–78, Lund, Sweden, 2012.

[21] L. Minku and X. Yao. An analysis of multi-objective evolutionary algorithms for training ensemble models based on different performance measures in software effort estimation. In *International Conference on Predictive Models in Software Engineering (PROMISE)*, Article No. 8, 10 pages, 2013.

[22] L. Minku and X. Yao. Ensembles and locality: Insight on improving software effort estimation. *Information and Software Technology (IST)*, 55(8):1512–1528, 2013.

[23] L. L. Minku and X. Yao. Software effort estimation as a multi-objective learning problem. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 22(4):Article No. 35, 32 pages, 2013.

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

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

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

[27] L. Song, L. Minku, and X. Yao. The impact of parameter tuning on software effort estimation using learning machines. In *International Conference on Predictive Models in Software Engineering (PROMISE)*, Article No. 9, 10 pages, 2013.

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

[29] I. Wieczorek and M. Ruhe. How valuable is company-specific data compared to multi-company data for software cost estimation? In *IEEE International Software Metrics Symposium (METRICS)*, pages 237–246, Ottawa, 2002.