PAPER

# LSA-X: Exploiting Productivity Factors in Linear Size Adaptation for Analogy-Based Software Effort Estimation

Passakorn PHANNACHITTA[†], *Nonmember*, Akito MONDEN[††a)], *Member*, Jacky KEUNG[†††], *Nonmember*, *and* Kenichi MATSUMOTO[†], *Member*

**SUMMARY** Analogy-based software effort estimation has gained a considerable amount of attention in current research and practice. Its excellent estimation accuracy relies on its solution adaptation stage, where an effort estimate is produced from similar past projects. This study proposes a solution adaptation technique named *LSA-X* that introduces an approach to exploit the potential of productivity factors, i.e., project variables with a high correlation with software productivity, in the solution adaptation stage. The *LSA-X* technique tailors the exploitation of the productivity factors with a procedure based on the Linear Size Adaptation (*LSA*) technique. The results, based on 19 datasets show that in circumstances where a dataset exhibits a high correlation coefficient between productivity and a related factor ($r \geq 0.30$), the proposed *LSA-X* technique statistically outperformed (95% confidence) the other 8 commonly used techniques compared in this study. In other circumstances, our results suggest using any linear adaptation technique based on software size to compensate for the limitations of the *LSA-X* technique.

***key words:*** *software development effort estimation, analogy, adaptation, productivity factor, empirical experiments*

## 1. Introduction

Analogy-based software development effort estimation (ABE) is a widely accepted estimation method. It derives an estimated effort value from the total amount of effort used on already completed similar software projects, following a hypothesis: *projects with similar characteristics will require similar amounts of effort to complete development* [1], [2]. Numerous empirical studies such as [2]–[4] have reported that ABE is an excellent method in terms of accuracy, as well as being intuitively easy to understand.

To establish a robust effort estimation process based on ABE, solution adaptation is one of the most imperative procedures. Its functions are to capture the degree of difference between the new project case and the similar cases or analogues, and to refine the estimated effort based on the degree of difference [5]. Solution adaptation is necessary because even the most similar past project available in a repository will deviate somewhat from the new case.

Manuscript received June 19, 2015.
Manuscript revised September 14, 2015.
Manuscript publicized October 15, 2015.

[†]The authors are with Graduate School of Information Science, Nara Institute of Science and Technology, Ikoma-shi, 630–0192 Japan.
[††]The author is with Graduate School of Natural Science and Technology, Okayama University, Okayama-shi, 700–8530 Japan.
[†††]The author is with Department of Computer Science, City University of Hong Kong, China.
a) E-mail: monden@okayama-u.ac.jp

DOI: 10.1587/transinf.2015EDP7237

Various adaptation techniques have been proposed in the literature such as the Linear Size Adaptation (*LSA*) technique [5], and many others [6]–[10]. However, despite being essential to analogy-based estimation, there is no generally accepted "best" solution adaptation technique. As suggested by a recent empirical assessment study by Azzeh [11], most techniques revisited in their studies only performed well on some specific selected datasets, while not performing well for others. Hence, we see that one of the major shortcomings in analogy-based estimation is the absence of a robust and stable solution adaptation technique that is readily suitable for various effort estimation datasets.

According to the observations addressed by Azzeh [11], the *LSA* and the Regression Towards the Mean (*RTM*) techniques can provide more accurate performance than any other techniques. These techniques adjust the effort based on productivity function: $Productivity = Effort/Software\ size$ [12]. In this study, we further explored the potential of exploiting productivity factors and propose a solution adaptation technique based on this exploitation. The proposed technique extends the adaptation phase of the *LSA* technique by exploiting productivity factors, project variables having the highest value of correlation with productivity, in the adaptation procedure. We named this technique e*X*tended *L*inear *S*ize *A*daptation (*LSA-X*). Specifically, *LSA-X* determines the productivity factor for a given dataset. It then calibrates the productivity value based on a linear extrapolation between productivity and the productivity factor. Finally, it applies the productivity function to the calibrated productivity and software size to produce the estimated effort value.

Our hypotheses behind the *LSA-X* are that (1) if a dataset exhibits a high correlation between productivity and its factor variable, we will be able to precisely refine the estimated productivity of the new case prior to adjusting its effort, (2) calibrating the productivity prior to adjusting the effort will produce a more robust estimate, as shown by the *RTM* technique [11], and (3) calibrating the productivity using linear extrapolation with its factor variable will generate a more significant estimate, in the same way that linear extrapolation was successfully performed between software size and effort in the *LSA* technique [11].

This paper is organized as follows. Section 2 provides the essential background such as the solution adaptation techniques of ABE. Section 3 introduces the *LSA-X* technique and explains its procedures. Section 4 describes

Copyright © 2016 The Institute of Electronics, Information and Communication Engineers

the evaluation methodology used in this paper, including the evaluation measures, datasets, and configuration parameters. Section 5 presents the results which were tested for statistical significance. Section 6 further discusses the results and threats to the validity of this study. Finally, we describe the conclusions of this paper in Sect. 7.

## 2. Background

### 2.1 Analogy-Based Software Development Effort Estimation (ABE)

ABE is a promising effort estimation method, commonly used in both the industry and research communities. We select ABE over other alternative model-based methods to study productivity factors because (1) it is widely used, widely studied, and consistently ranked among the best-performing methods in terms of accuracy [1], [5], [11], [13], and (2) various approaches to improve its accuracy by exploiting productivity measures have been proposed as ways to refine the estimates in the solution adaptation stage.

Using ABE to estimate the software development effort involves a 4-stage case-based reasoning process [14], consisting of: **Retrieve** the project cases most similar to the new case, **Reuse** the information from the retrieved past cases to propose a solution to the new case, **Revise** the proposed solution to better adapt to the new case, and **Retain** the solved case for future problem solving.

Figure 1 shows the computing processes of ABE that are commonly implemented as a software development effort estimation framework. As shown in Fig. 1, the solution adaptation process is the final stage that captures the difference between the retrieved project cases and the new case, and revises the estimated effort value based on the degree of difference.

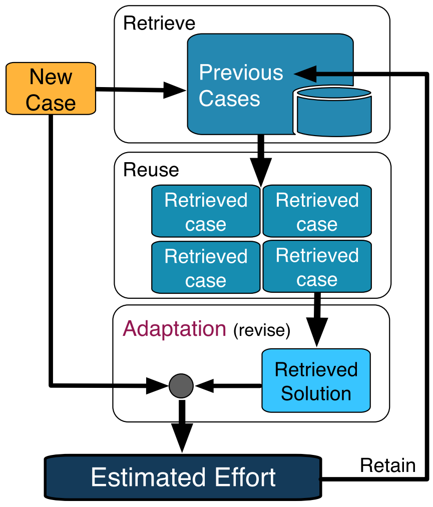


**Fig. 1** The estimation process of analogy-based estimation

### 2.2 Solution Adaptation Techniques for ABE

In a study by Azzeh [11], 8 commonly used solution adaptation techniques were replicated and evaluated to provide an empirical assessment of their effectiveness. Azzeh [11] reported that all the revisited techniques were still a long way from reaching the optimal solutions. Nonetheless, we found 2 interesting observations made for estimation techniques that exploited productivity as a function of $Productivity = Effort/Software\ size$ in Azzeh's study, which are:

- A technique named Linear Size Adaptation (*LSA*), which adjusts the effort using a linear extrapolation between size and effort, often generated significantly better (95 % confidence) estimate than any other techniques.
- A technique named Regression Toward the Mean (*RTM*), which calibrates the productivity value across project cases in a dataset by using a statistical phenomenon called *regression towards the mean*, was most frequently selected as most accurate technique for moderate size datasets with a coherent control group.

We speculate that an exploitation of either software size or productivity was the major factor in the performance achieved by the *LSA* and *RTM* techniques. These 2 findings motivated us to further explore exploitation of productivity in the solution adaptation stage of the ABE by examining its factor variables.

Table 1 lists and summarizes the 8 adaptation techniques compared in this study. In Table 1, *LSA*, *MSA* and *RTM* are the techniques that utilize the software size. Due to space limitations in this paper, we only provide a detailed explanation for these 3 techniques, along with a brief explanation of the other 5 techniques.

**Linear Size Adaptation (LSA)** is based on linear extrapolation of the size differences between a new case and its analogues, to determine an effort value for the new case based on the effort values of the analogues [5]. In this calculation, the software size variable may be measured in terms such as *adjusted Function Points*, *raw Function Points*, or *lines of code*. Walkerden and Jeffery [5], based on their

**Table 1** The 8 adaptation techniques selected in this study.

| Abbrev. | Adaptation Techniques | Adjustment function | Adjustment feature |
|---|---|---|---|
| 1 UAVG [15] | Unweighted mean of the $k$ analogues | Mean | Effort |
| 2 IRWM [16] | Inverse-ranks weighted mean of the $k$ analogues | Mean | Effort |
| 3 LSA [5] | Linear size adaptation | Linear | Software size |
| 4 MSA [6] | Multiple size adaptation | Linear | Software size(s) |
| 5 RTM [7] | Regression towards the mean | Linear | Software size |
| 6 AQUA [8] | Similarity-based adaptation | Linear | All features |
| 7 GA [9] | Adaptation based on Genetic algorithm | Linear | All features |
| 8 NNet [10] | Non-linear adaptation based on Neural network | Non-linear | All features |

observation, suggested that an effort adaptation using only a software size variable is rational, because any size variables are always strongly correlated with the effort. Furthermore, adjusting the effort using a size variable also allows an estimation to scale the estimated effort value between pairs of analogues based on differences in size. Equation (1) depicts the formula of the *LSA* technique.

$$Eff(P_{new}) = \frac{SS(P_{new}) * Mean(\ Eff(P_{analogs})\ )}{Mean(\ SS(P_{analogs})\ )}, \tag{1}$$

where *SS* indicates a single software size variable, and *Eff* indicates an effort value. This equation can be explained as a function of *Productivity = Effort/Size*, where $Pr(P_{analogs})$ is the productivity values of the analogue projects:

$$Eff(P_{new}) = SS(P_{new}) * Mean(\ Pr(P_{analogs})\ ), \tag{2}$$

**Multiple Size Adaptation (MSA)** extends the *LSA* technique to handle the case where the software size is elaborated with multiple arbitrary attributes, such as in web application development [6]. *MSA* aggregates multiple size variables using the mean and applies the mean software size to Eq. (1). This aggregation of size variables can be considered as calibrated productivity since this function retains the ratio between outputs and inputs [12].

**Regression Toward the Mean (RTM)** calibrates the productivity of a project case to be more consistent with other project cases [7]. The productivity calibration follows the statistical phenomenon known as *regression toward the mean*, which states that any extreme instance found on its first measurement will tend to move toward its population mean on its second measurement. In this estimation process, the productivity of each project is adjusted towards the mean productivity of the projects in the same coherence group. Project cases are divided into groups using a single categorical variable selected prior to the estimation. The adjusted productivity is then calculated for the estimated effort following *Productivity = Effort/Software size*. The formula of the *RTM* technique is depicted in Eq. (3):

$$\begin{aligned} Eff(P_{new}) = SS(P_{new}) * \Big(&Mean(\ Pr(P_{analog})\ ) \\ &+ (M - Mean(\ Pr(P_{analog}))\ ) \times (1-r)\Big), \end{aligned} \tag{3}$$

where *M* is the mean productivity of the projects in the same coherence group, and *r* is the correlation between the productivity of the project analogues and the actual productivity of the projects in the same group.

For the remaining 5 techniques in Table 1, *UAVG* is the fundamental technique used in the basic *ABE* framework [1], [4]. It aggregates the effort values selected from the *k* analogues using a simple unweighted mean. *IRWM* applies different weight values to different project analogues, assuming that project analogues that are closer to the new case are more important [16]. *AQUA* adjusts the estimated effort by aggregating the degree of similarity between pairs of project features of the new case and its analogues. The aggregation formula involves the sum of the product of the normalized similarity degree between all the project features being selected for the estimated effort. *GA* adopts a genetic algorithm to adjust the retrieved effort values based on the similarity degrees between all pairings of project cases. Finally, *NNet* trains artificial neural networks to capture the degree of difference of the project features between pairings of project cases, and applies the captured degree of difference to the retrieved effort from the *k* analogues.

## 3. The Proposed LSA-X Technique

From Eq. (1), the *LSA* technique estimates the term *Effort/Size* as the objective variable of an ABE system instead of the effort. It then applies a linear adjustment based on a software size variable of the new case to produce its final estimated effort value.

As an extension to the *LSA* technique, the proposed eXtended Linear Size Adaptation (*LSA-X*) technique introduces a procedure to utilize productivity (*Pr*) and an influential factor variable (*PrFactor*) in the solution adaptation stage of the analogy-based estimation. *LSA-X* adds one more adaptation step to the *LSA* technique by calibrating the productivity of the retrieved similar project cases before adjusting the effort using a software size. Specifically, *LSA-X* applies a basic analogy-based estimation framework to estimate the term *Pr/PrFactor* as well as *Effort/Software size* for the new case using the *PrFactor* and software *size* of the new case to produce the required effort value.

### 3.1 The Procedure

The *LSA-X* technique has three adaptation steps:

**Step 1** is a search for the *PrFactor*. We define the *PrFactor* as any single continuous project variable of a dataset that has the highest positive degree of Pearson correlation with the productivity (*Pr*). This single criteria is extended from the essence of the *LSA* technique, where software size is the variable of choice because it commonly exhibits the highest correlation to the effort, as empirically observed in the study by Walkerden and Jeffery [5]. In addition, since the Pearson correlation assumes a normal distribution, we apply a logarithmic transformation to all the continuous variables prior to perform the correlation test, as suggested by Kitchenham and Mendes in [17].

In **Step 2**, after the *PrFactor* is identified, the *Pr* of the new case is calibrated using linear adjustment:

$$\begin{aligned} Pr'(P_{new}) = &PrFactor(P_{new}) \\ &\times \left(\frac{Mean(\ Pr(P_{analog})\ )}{Mean(\ PrFactor(P_{analog})\ )}\right) \end{aligned} \tag{4}$$

where $Pr'$ is the calibrated productivity.

In **Step 3**, *LSA-X* adjusts the term $Pr'$ using a software size variable from the new case. This is done by replacing the

| Case | Size | Required Reliability | Lang | Arch | Actual Effort |
|---|---|---|---|---|---|
| $P_{new}$ | 100 | 4/10 | Java | Stand Alone | **100** |
| $P_{analog}$ | 200 | 8/10 | Java | Stand Alone | 400 |

| Estimated Effort | | |
|---|---|---|
| *UAVG* | *LSA* | *LSA-X* |
| 400 | $400 \times \frac{100LOC}{200LOC} = 200$ | $400 \times \frac{100LOC}{200LOC} \times \frac{0.4}{0.8} = \mathbf{100}$ |

**Fig. 2** An example comparison

term *Pr* in Eq. (2) of the *LSA* technique with the $Pr'$, calculated in Step 2. Equation (5) depicts the third adaptation step of *LSA-X*.

$$Eff(P_{new}) = SS(P_{new}) \times Pr'(P_{new}) \tag{5}$$

### 3.2 The Rationale Behind the LSA-X Technique

Compared to the basic *UAVG* technique, the *LSA* technique generally produces a higher accuracy [5], [6], [16] because it can capture and exploit the degrees of difference between project cases with similar features, but different software sizes. *LSA* applies the degree of the difference in software size to the estimated effort value and scales the estimated effort to match the characteristics of the new case.

The proposed *LSA-X* technique can further identify and exploit the productivity differences between a new project case and its analogues by first adjusting the productivity, followed by the software size. Figure 2 provides an example case where adjusting both software size and productivity using the *PrFactor* allows *LSA-X* to be more accurate than either *LSA* or *UAVG*.

In this example, the new case is an expected 100 LOC Java standalone program with a required reliability of 4 on an ordinal scale of 10 levels. We also know that the developers spent 100 man hours to complete the new case. After performing a search of the past project cases, the most similar project found was a case with a 200 LOC Java standalone program, and a required reliability level of 8/10. This analogue consumed 400 man-hours to complete its development.

Applying the *UAVG* technique to this example, the ABE framework only reuses the effort value from the analogue case and estimates the new case will take 400 man-hours as the required amount of effort for its development. On the other hand, an estimated effort adjusted using the *LSA* technique results a more accurate estimate. From this example, the *LSA* technique captures the information that the new case is about 50% smaller in software size. *LSA* then scales down the estimated effort value to 200 man-hours, which is more accurate than the *UAVG* result.

Following the adaptation mechanism of *LSA* that the effort value of the analogue should be adjusted by a software size ratio to match the characteristics of the new case, the proposed *LSA-X* technique further adjusts the estimated effort value by exploiting one additional variable. In this example, *LSA-X* captures the required software reliability as a *PrFactor* for this dataset, and the new case has a required level of reliability that is 50% less than the closest analogue. The *LSA-X* technique adds one more adaptation step to adjust the effort value based on both the software size and in this example the required software reliability. *LSA-X* thus estimates the required effort for the new case as 100 man-hours, which is more accurate.

## 4. Evaluation Methodology

This section describes the error measures, datasets, feature selection methods, evaluation procedure, and configuration parameters selected and used in this study to evaluation the performance of the proposed *LSA-X* technique in comparison to other solution adaptation techniques for ABE.

### 4.1 Five Error Measures

Table 3 summarizes the 5 error measures we used in our experiments. These 5 error measures were selected according to the suggestion by a study of Foss et al. [26], in which the validity of numerous error measures commonly used in the software development effort estimation literature were systematically evaluated. These 5 measures are considerably more robust and valid than many others such as the well-known *MMRE* and *Pred*(25). From the 5 measures selected, 2 are the absolute errors based on the absolute residual, and the remaining 3 are variants of the standard deviation: Standard Deviation *(SD)*, Relative Standard Deviation *(RSD)*, and Logarithmic Standard Deviation *(LSD)*.

### 4.2 19 Datasets and Their *PrFactors*

We used 19 software effort estimation datasets in our experiments. All of these are available in the tera-PROMISE software engineering repository (Accessible through [18]), where the datasets are made available and commonly used in empirical software engineering studies. To maximize the validity of our study, these 19 datasets were taken from the list of 20 datasets which is the highest number of datasets that appeared in software development effort estimation literature [4], [13]. We decided to exclude one dataset, namely Telecom, because it does not contain any software size variables, which are required by the *LSA*, *MSA*, *RTM* and the proposed *LSA-X* techniques.

Table 2 presents the details of the experimental datasets sorted by the value of $r(Pr, PrFactor)$, along with the details of the *PrFactor* selected for each dataset. Even though the characteristics of the 19 datasets were greatly diversified, we can categorize the variables selected as the *PrFactor* in three groups, all of which are known to be influential factors of *Pr*:

- Size, complexity and constraints of the software project, e.g., Rely, Time, and Flex;
- Developers' experience, e.g., TeamExp and Lexp;
- Tool usage, e.g., T08, and Tool.

**Table 2** The 19 benchmark datasets ($F$ = features, $N$ = project cases, $Pr$ = productivity, $PrFactor$ = productivity factor variable)

| Dataset | F | N | Dataset descriptions | PrFactor | PrFactor Description | r(Pr, PrFactor) |
|---|---|---|---|---|---|---|
| China | 12 | 499 | Projects from multiple Chinese companies [18] | Interface | External interface added | 0.01 |
| Desharnais-2 | 11 | 25 | Desharnais projects with language 2 [1] | Transactions | Number of the logical transactions | 0.05 |
| Kemerer | 7 | 15 | A small data set from Large Business [19] | Ksloc | Source lines of code | 0.09 |
| Desharnais | 8 | 48 | The well-known Canadian software project [1] | Envergure | Complexity adjustment factors | 0.11 |
| Miyazaki94 | 12 | 81 | Software project developed in COBOL [20] | File | Number of different record formats | 0.16 |
| Desharnais-1 | 11 | 46 | Desharnais projects with language 1 [1] | Envergure | Complexity adjustment factors | 0.21 |
| Finnish | 7 | 38 | Data from different companies in Finland [21] | Co | N/A | 0.30 |
| Maxwell | 27 | 62 | Projects from commercial banks in Finland [22] | T08 | Impact of tools | 0.37 |
| Desharnais-3 | 11 | 10 | Desharnais projects with language 3 [1] | TeamExp | Team experience | 0.39 |
| Albrecht | 8 | 24 | Completed projects from IBM in 70s [23] | RawFPcounts | Size in terms of raw Function Points | 0.59 |
| Nasa93-c2 | 22 | 37 | Nasa93 projects from center 2 [24] | Virt | Machine volatility | 0.54 |
| Cocomo81-e | 17 | 28 | Embedded project type in Cocomo81 [12] | Lexp | Language experience | 0.59 |
| Cocomo81 | 18 | 63 | Nasa software projects [12] | Time | Time constraint for cpu | 0.62 |
| Nasa93 | 23 | 93 | Nasa software projects [24] | Time | Time constraint for cpu | 0.65 |
| Cocomo-sdr | 12 | 24 | Projects from various companies in Turkey [25] | Flex | Development Flexibility | 0.66 |
| Cocomo81-s | 17 | 11 | Semidetached project type in Cocomo81 [12] | Rely | Required software reliability | 0.69 |
| Nasa93-c5 | 22 | 48 | Nasa93 projects from center 5 [24] | Time | Time constraint for cpu | 0.75 |
| Nasa93-c1 | 22 | 12 | Nasa93 projects from center 1 [24] | Rely | Required software reliability | 0.79 |
| Cocomo81-o | 17 | 24 | Organic project type in Cocomo81 [12] | Tool | Use of software tools | 0.81 |

**Table 3** A summary of the 5 error measures used in this study. ($x_i$ is the actual effort, $\hat{x}$ is the estimated effort, $N$ is the total number of project cases, $SS$ is a single software size variable, and $s$ is the estimated variance of $ln(x_i/\hat{x}_i)$).

| | Abbrev. | Error measure | Formula |
|---|---|---|---|
| 1 | MAR | Mean of Absolute Residual | $mean(\lvert x - \hat{x}\rvert)$ |
| 2 | MdAR | Median of Absolute Residual | $median(\lvert x - \hat{x}\rvert)$ |
| 3 | SD | Standard Deviation | $\sqrt{\frac{\sum_{i=1}^{N}(x_i-\hat{x}_i)^2}{N-1}}$ |
| 4 | RSD | Relative Standard Deviation | $\sqrt{\frac{\sum_{i=1}^{N}\left(\frac{(x_i-\hat{x}_i)^2}{SS_i}\right)^2}{N-1}}$ |
| 5 | LSD | Logarithmic Standard Deviation | $\sqrt{\frac{\sum_{i=1}^{N}\left(ln(\frac{x_i}{\hat{x}_i})-\left(-\frac{s^2}{2}\right)\right)^2}{N-1}}$ |

In the case that a *PrFactor* was selected with a very low value of $r(Pr, PrFactor)$, e.g. $r(Pr, PrFactor) < 0.30$, such as the first 6 data sets in Table 2, we speculate that these datasets may not contain any real *PrFactors*. This may be due to their data collection or being a large and massively divergent dataset, such as the China dataset, which consists of numerous project cases collected from various companies [27]. We therefore hypothesize that the proposed *LSA-X* technique will not perform well for these datasets since the *PrFactor* seems not to be useful.

## 4.3 Four Feature Subset Selection Methods

Feature subset selection is one of the most influential factors determining the performance of an ABE framework [4], [28]. To avoid the issue of unstable results and to improve generalization of the results, we adopted 4 feature selection methods in this study, all of which are often adopted in software effort estimation literature [4], [13]:

**All** selects all the features in a dataset.

**Sfs** (Sequential forward selection [29]) is a greedy algorithm that adds features one by one into the feature subset candidate until there is no improvement when adding any of the remaining features. The criteria used to evaluate the improvement of **Sfs** was the same as used in a study by Keung et al. [4], which is the default MATLAB's *objective* function. This function reports the mean-square error of a simple linear regression on the training set.

**Swr** (Stepwise regression [29]) iteratively adds or removes features from the feature set based on the statistical significance of a multilinear model in explaining the objective variable.

**Pca** (Principal component analysis [8]) uses a transformation function to map the entire dataset into a new lower-dimensional space, where remaining values are uncorrelated but they still retain the same variance as in the original dataset.

## 4.4 Evaluation Procedure

To enable and ease the reproduction of the experiments, we adopted a procedure that was used in many recent prominent studies [4], [8], [13], [30]. In this study, the method used to sample the training/test instances is based on leave-one-out approach [31] (a.k.a. Jack-knifing, where each software project case is the test instance of a model built from all the remaining cases). The leave-one-out is our method of choice, because (1) it is widely used [4], [8], [13], [30], (2) it does not depend on any randomization, so that an evaluation over the same dataset will have the exact same result in every single run, and (3) a recent study by Kocaguneli [32] compared leave-one-out with the widely used alternative N-way cross validation method, and suggested that the leave-one-out approach often generates lower estimation bias and is more robust for small and medium-sized datasets [13], [33]

```
1: if $Wilcoxon(Err_i, Err_j, 0.95)$ says they are the same then
2:     $tie_i = tie_i + 1$
3:     $tie_j = tie_j + 1$
4: else
5:     if better($Err_i$,$Err_j$) then
6:         $win_i = win_i + 1$
7:         $loss_j = loss_j + 1$
8:     else
9:         $win_j = win_j + 1$
10:        $loss_i = loss_i + 1$
11:    end if
12: end if
```

**Fig. 3** Computing the win-tie-loss statistic between adaptation techniques $i$ and $j$ on a single error measure *Err*. A better performance of all the 5 error errors is indicated by a lower error values.

(i.e. less than 100 cases). Over 90% of the available software effort estimation datasets in the tera-PROMISE repository are this size [18].

The overall performance of an adaptation technique is summarized from 5 error measures: *MAR*, *MdAR*, *SD*, *LSD*, and *RSD* using the win-tie-loss statistic [4], [13], [30]. The *win-tie-loss* statistic adopts a Wilcoxon rank-sum hypothesis test (95% confidence) [34] based on a confidence interval of the *p-hat* metric [35]. This variant of the statistical test was selected because the Wilcoxon rank-sum test is a robust test method for situations where the samples do not have an underlying Gaussian distribution, and the confidence interval of *p-hat* metric was suggested by Zimmerman [36] as a more stable variant of the Wilcoxon test.

Figure 3 explains the computing procedure of the win-tie-loss statistic. With this statistic, the results are measured through three counters, wins, ties, and losses. The statistic counts the results of pairwise comparisons of all pairs of techniques for each error measure. So the computing procedure shown in Fig. 3 tests one pair of techniques for one error measure, and is then repeated for all pairs of techniques and all error measures. Thus, if a technique $i$ totally outperforms a technique $j$ in terms of all 5 error measures, its total wins will increment by 5, while the total losses of technique $j$ will increase by 5.

A summation of total *wins*, *losses*, or *wins - losses* is commonly used as a trustworthy procedure to compare the software effort estimators [4], [37], [38]; however, there is no generally accepted method among these three interpretations [13], [34]. In this study, we used only the total *wins* to report the detailed results where all the three interpretations cannot be presented due to space limitations in this paper. For more summarized results, we presented the comparison results based on all terms of *wins*, *losses*, and *wins - losses*. Furthermore, the *wins*, *losses*, and *wins - losses* are iterated on each single feature subset selection method individually. Then, the median value of the same summations were calculated across the 4 feature subset selection methods to present the overall performance of a solution adaptation technique. The median was taken here to avoid bias possibly introduced by any single specific feature selection method.

## 4.5 Configuration Parameters

In this study, all the ABE variants were implemented using MATLAB. In the experiments, the baseline ABE framework that all the adaptation techniques used is a variant of the standard *ABE0-kNN* framework [4]. The similarity between project cases is based on the Euclidean function [2]. We normalized all the continuous variables in the interval between 0 and 1 to assure the equal influence of all the individual continuous features. The normalization for a feature vector $i$ was produced by subtracting the minimum value of the vector from all the attribute values, then dividing result by value of the maximum minus the minimum values of $i$). This procedure is widely used in effort estimation studies such as in [4], [8], [37], [39], [40].

For the choice of the number of analogues ($k$), we adopted a more optimal approach suggested by Baker [38], in which the value of $k$ is tuned to be the best fit with any given training/test splits. Thus, a different training/test split of a dataset may produce a different $k$ value. This approach is considerably more intuitive than fixing a single $k$ value for the entire dataset, which is commonly used in the software effort estimation literature [5], [6], [10]. In addition, selection of the $k$ value in this way also allows us to omit the sensitivity analysis of the effectiveness of the adaptation techniques regarding the number of analogues, because the accuracy is no longer subject to an arbitrary assignment of a single $k$ value.

Some models such as *RTM* and *NNet* require specific configuration parameters. The parameters we configured for these models are given below.

**LSA**: For this technique, the single size variable to be selected is based on experimentations. For each dataset, the selected size variable is the one that exhibits the highest correlation coefficient with the effort. Since there is no generally accepted method to select a single size variable, we adopted this criteria to conform with the essence of the proposed study by Walkerden and Jeffery [5].

**RTM**: *RTM* has two specific parameters: the single size variable and the variable to represent the coherence groups. For the size variable, we used the same criteria as used in the *LSA* technique. For the second variable, we carried out a literature review on the studies that adopted or replicated the *RTM* technique in the area of software effort estimation. Our review found that existing studies [7], [11] suggested using a categorical variable to represent the coherence groups. However; none of the existing studies discussed the selection criteria when there are more than one categorical variable present in a dataset. We adopted simple criteria as follows:

- In cases where no categorical variable existed in a dataset, we let the entire dataset be a single coherent group.
- In cases where that there is one or more categorical

variable, the variable of choice is the one that generates coherence groups with the minimum sum of pairwise difference of the effort, calculated across all the divided coherence groups.

**GA**: In this study, the optimization criteria for the genetic algorithm is based on the MAR error measure. This is different from the proposal studied by Chiu and Huang [9] which relied on MMRE and PRED. Earlier in this section, we discussed why these two measures are no longer believed to be the trustworthy measures.

**NNet**: We trained the networks using the "trainscg" MATLAB function because it enables acceleration of the experiments through parallel computation. We configured the networks with 2 hidden layers as commonly done in several recent studies such as in [13].

## 5. Results

Table 4 shows the results of the experimental comparison of the 9 adaptation techniques. In this table the *win-tie-loss* statistic are summarized by the number of *wins*. Each entry in Table 4 is the median of the total sum of *wins* aggregated from the 4 feature selection methods and the 5 error measures across the leave-one-out experiments. Hence the maximum number of *wins* for an entry in Table 4 is $\big((9 \text{ techniques} \times 4 \text{ feature selectors}) - 1 \text{ self}\big) \times 5 \text{ error measures} = 175$. This is the maximum number of *wins* for a variant that pairs one technique and one feature selection method. The highlighted entries in Table 4 show the technique with the maximum number of *wins* for the dataset in that row. The results for each dataset, the rows of Table 4, are sorted by the Pearson correlation strength between productivity and productivity factor ($r(Pr, PrFactor)$).

The results of Table 4 show that the *LSA-X* technique performed best in terms of *wins* for 9 of the 19 datasets, *MSA* was next, with 5/19. *RTM* and the *GA* were the best for 2 datasets, while *NNet* performed the best for 1 dataset. According to the results of these preliminary experiments, *LSA-X* yielded the best general performance.

When we look closely at the $r(Pr, PrFactor)$ column in Table 4, we can divide the datasets into 2 bands based on $r(Pr, PrFactor)$ with regard to the accuracy achieved by the *LSA-X* technique:

**Band 1:** datasets with $r(Pr, PrFactor) < 0.30$;

**Band 2:** datasets with $r(Pr, PrFactor) \geq 0.30$.

*LSA-X* was the method of choice for datasets with $r(Pr, PrFactor)$ equal to or greater than 0.30. In this band, *LSA-X* had the best performance for 9 out of the 13 datasets. For 3 of the other 4 datasets in this band, it also performed almost well as the best performers. On the other hand, *LSA-X* performed poorly for the datasets with $r(Pr, PrFactor)$ less than 0.30. In these cases, the *MSA* technique performed well for 3 out of the 6 datasets, with *RTM*, *AQUA*, and *Nnet* as the best performing techniques for each of the remaining 3 datasets.

To examine which solution adaptation techniques are best for each band of datasets, we aggregated the *wins*, *losses*, and *wins - losses* for each band using the mathematical mean. Table 5 presents the aggregation results. The highlighted entries in Table 5 show the best techniques in each band in terms of *wins*, *losses*, and *wins - losses*. Presented in this way, the results not only provide a ranking of adaptation techniques across the two bands of datasets, but also show the stability of teach technique as measured by the

**Table 4** Win-tie-loss results from (4 feature selection methods x 5 error measures) of Leave-one-out experiments. The highlighted entires indicate the estimators with best performance in terms of *wins*.

| Datasets | wins | | | | | | | | | $r(Pr, PrFactor)$ |
|---|---|---|---|---|---|---|---|---|---|---|
| | UAVG | IRWM | LSA | MSA | LSA-X | RTM | AQUA | GA | Nnet | |
| China | 32.50 | 35.50 | 37.75 | 37.75 | 5.00 | 34.75 | 18.00 | 32.25 | 43.25 | 0.01 |
| Desharnais-2 | 2.50 | 3.25 | 25.00 | 30.50 | 0.00 | 31.75 | 1.00 | 4.25 | 1.25 | 0.05 |
| Kemerer | 1.00 | 1.00 | 6.50 | 7.00 | 1.00 | 6.50 | 1.00 | 1.00 | 0.75 | 0.09 |
| Desharnais | 12.75 | 14.00 | 64.00 | 64.00 | 33.75 | 50.00 | 3.75 | 13.75 | 1.50 | 0.11 |
| Miyazaki94 | 10.00 | 1.75 | 3.00 | 3.75 | 1.50 | 3.50 | 1.25 | 9.50 | 0.00 | 0.16 |
| Desharnais-1 | 5.75 | 3.75 | 8.00 | 10.75 | 4.25 | 10.00 | 0.25 | 5.75 | 1.50 | 0.21 |
| Finnish | 5.00 | 3.50 | 18.25 | 18.25 | 22.25 | 11.25 | 1.00 | 5.00 | 3.00 | 0.30 |
| Maxwell | 15.00 | 12.25 | 27.75 | 27.75 | 52.00 | 14.50 | 8.00 | 15.00 | 10.75 | 0.37 |
| Desharnais-3 | 4.75 | 3.25 | 7.00 | 7.00 | 9.75 | 8.50 | 0.00 | 4.75 | 0.25 | 0.39 |
| Albrecht | 0.00 | 0.00 | 0.00 | 0.25 | 0.50 | 0.00 | 0.00 | 0.00 | 0.00 | 0.59 |
| Nasa93-c2 | 23.75 | 24.25 | 61.00 | 61.00 | 40.25 | 53.25 | 28.50 | 23.75 | 9.75 | 0.54 |
| Cocomo81-e | 8.00 | 12.50 | 45.25 | 45.25 | 57.50 | 66.50 | 9.00 | 8.00 | 3.00 | 0.59 |
| Cocomo81 | 12.25 | 16.25 | 93.25 | 93.25 | 89.00 | 76.75 | 15.25 | 12.25 | 6.25 | 0.62 |
| Nasa93 | 27.00 | 35.75 | 84.25 | 84.25 | 85.00 | 72.50 | 35.00 | 29.25 | 15.25 | 0.65 |
| Cocomo-sdr | 16.75 | 14.50 | 27.00 | 27.00 | 45.25 | 25.75 | 13.25 | 7.00 | 10.50 | 0.66 |
| Cocomo81-s | 10.25 | 12.25 | 31.25 | 31.25 | 32.75 | 22.00 | 10.75 | 12.75 | 9.25 | 0.69 |
| Nasa93-c5 | 10.50 | 13.25 | 63.50 | 63.50 | 76.75 | 61.50 | 9.75 | 10.75 | 3.75 | 0.75 |
| Nasa93-c1 | 11.00 | 10.50 | 5.25 | 5.25 | 9.00 | 11.00 | 12.75 | 13.50 | 3.00 | 0.79 |
| Cocomo81-o | 10.75 | 11.25 | 34.50 | 34.50 | 34.75 | 17.25 | 3.75 | 10.75 | 5.00 | 0.81 |

**Table 5** A summary of the results of Table 4 in the 2 bands of datasets. The highlighted entries show the best-performing techniques.

| Datasets | Measures | UAVG | IRWM | LSA | MSA | LSA-X | RTM | AQUA | GA | Nnet |
|---|---|---|---|---|---|---|---|---|---|---|
| Band 1 ($r < 0.30$) | wins | 10.75 | 9.88 | 24.04 | 25.63 | 7.58 | 22.75 | 4.21 | 11.08 | 8.04 |
| | losses | 8.67 | 5.92 | 1.88 | 1.88 | 40.67 | 1.38 | 28.33 | 9.13 | 26.13 |
| | wins-loses | 2.08 | 3.96 | 22.17 | 23.75 | −33.08 | 21.38 | −24.13 | 1.96 | −18.08 |
| Band 2 ($r \geq 0.30$) | wins | 11.92 | 13.04 | 38.33 | 38.35 | 42.67 | 33.90 | 11.31 | 11.75 | 6.13 |
| | losses | 33.17 | 30.79 | 1.87 | 1.87 | 1.00 | 3.69 | 37.29 | 33.98 | 63.75 |
| | wins-losses | −21.25 | −17.75 | 36.46 | 36.48 | 41.67 | 30.21 | −25.98 | −22.23 | −57.62 |

**Table 6** A comparison of the 3 most accurate adaptation techniques in terms of the 4 feature subset selection methods

| Datasets | Measures | MSA | | | | LSA-X | | | | RTM | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | All | Sfs | Swr | Pca | All | Sfs | Swr | Pca | All | Sfs | Swr | Pca |
| Band 1 ($r < 0.30$) | wins | 22.17 | 26.50 | 28.17 | 25.67 | 9.17 | 8.67 | 2.50 | 10.00 | 20.17 | 21.00 | 24.50 | 25.33 |
| | losses | 1.00 | 1.00 | 1.50 | 4.00 | 38.33 | 39.00 | 49.00 | 36.33 | 1.00 | 2.00 | 1.50 | 1.00 |
| | wins-loses | 21.17 | 25.50 | 26.67 | 21.67 | −29.17 | −30.33 | −46.50 | −26.33 | 19.17 | 19.00 | 23.00 | 24.33 |
| Band 2 ($r \geq 0.30$) | wins | 39.92 | 39.85 | 33.62 | 40.00 | 43.54 | 42.77 | 41.23 | 43.15 | 36.00 | 36.62 | 35.15 | 27.85 |
| | losses | 2.38 | 1.31 | 0.69 | 3.08 | 0.23 | 0.23 | 2.31 | 1.23 | 2.31 | 2.77 | 1.08 | 8.62 |
| | wins-losses | 37.54 | 38.54 | 32.92 | 36.92 | 43.31 | 42.54 | 38.92 | 41.92 | 33.69 | 33.85 | 34.08 | 19.23 |

*win-tie-loss* statistics. For example, a technique having high *wins* and low *losses* is more stable than another technique with very high *wins* but also high *losses*.

Table 5 provides strong evidence that the *LSA-X* technique consistently performed best for datasets with $r(Pr, PrFactor)$ equal to or greater than 0.30, as its *wins*, *losses*, and *wins-losses* in this table are better than all other techniques selected in this study. Furthermore, its averaged *losses* almost reached 0, the the absolute best possible performance in terms of *losses*. On the other hand, the datasets with $r(Pr, PrFactor)$ less than 0.30 are clearly a limitation of the *LSA-X* technique. For this band of datasets, Table 5 shows a very close performance among the *LSA*, *MSA* and *RTM* techniques. *MSA* is slightly better than the other two techniques. Hence, for this band of data sets, we suggest the use of any of these techniques to compensate for the limitation of *LSA-X*, and would suggest *MSA* if we have to select only one technique.

The *win-tie-loss* results of Table 4 and Table 5 calculated used the mean of the accuracy results to demonstrate the generalized performance of each adaptation technique. To examine more specific results regarding the use of different feature subset selection methods, Table 6 shows the performance in regard to all 4 feature selection methods. Due to space limitations, Table 6 only presents the top 3 solution adaptation techniques ranked by performance as in Table 5. These specific results indicate that for the datasets with $r(Pr, PrFactor)$ greater than 0.30, *LSA-X* performed best when the feature subset selection was selected by the **All** method, followed by **Sfs**, **Pca**, and **Swr** in order of *wins - losses*. On the other hand, the results in Table 6 for the datasets in band 1 suggest using the **Swr** to select the feature subset for the *MSA* technique as the best effort estimation variant.

## 6. Discussion

This study introduces a solution adaptation technique named *LSA-X* for analogy-based software effort estimation. The *LSA-X* adds one more adaptation step to the Linear Size Adaptation (*LSA*) technique by exploiting the productivity factor variables (*PrFactor*) in its effort adjustment procedure. The highlighted benefits of the *LSA-X* technique are described below.

**Benefit 1:** We can see from the highlighted entries in Table 4 that *LSA-X* achieved the best generalized performance. Even though we later found that the *LSA-X* was consistently less accurate than other techniques with datasets that exhibit a value of $r(Pr, PrFactor)$ below 0.30, such case are a minority of datasets and may be unlikely to arise in practice.

**Benefit 2:** *LSA-X* does not require a high computational power. Compared to more complex adaptation techniques such as the *GA* and *NNet* techniques, *LSA-X* is significantly more straightforward, such that even a spreadsheet application can handle the necessary computation.

**Benefit 3:** It is easy to decide when to use *LSA-X*. As we observed, a Pearson correlation coefficient between *Pr* and *PrFactor* can simply determine the accuracy of the *LSA-X* technique:

- *LSA-X* is a very robust technique for datasets with $r(Pr, PrFactor)$ equal to or greater than 0.30;
- *LSA-X* is consistently inaccurate for datasets with $r(Pr, PrFactor)$ less than 0.30.

Hence, we recommend the *LSA-X* technique as the adaptation technique of choice in circumstances where a dataset exhibits $r(Pr, PrFactor)$ equal or greater than 0.30.

### 6.1 And Why does it Work?

To provide a clear justification as to why exploiting the productivity factors using *LSA-X* is an improvement, further quantitative analysis may be required. However, this section discusses some possible reasons.

The first reason concerns the exploitation of productivity. The benefit of productivity exploitation in predictive modeling is well recognized in various research areas such as Economics, because productivity is one of the important factors to analyze to improve the throughput of a work process [41], [42]. Among the existing solution adaptation techniques that consider the degrees of difference of both software size and productivity between the new case and its analogues, only the *LSA-X*, *MSA* and *RTM* techniques also calibrate the productivity values in their adaptation procedures. Our results showed the *MSA* and *RTM* techniques were also high performance techniques.

The second reason concerns the use of the *PrFactors* in addition to productivity. This is possibly the main reason that *LSA-X* outperforms all the other techniques when *PrFactors* are useful, i.e., $r(Pr, PrFactors) \geq 0.30$. Based on our observations, productivity is considered the second most influential factor on the effort next to the software size. That is, when the influence of the software size is removed from the dataset, productivity then becomes the most dominant factor of the effort. Therefore, when considering adaptation, we first need to consider the difference in software size among analogues, and next, we need to consider the difference in productivity among analogues. The use of *PrFactors* plays the main role in better measuring and adapting productivity differences among analogues.

Specifically, the use of *PrFactors* allows the *LSA-X* technique to focus the productivity adjustment based only on the difference in *PrFactors* between the new case and a few of its analogues, rather than requiring the adjustment to be based on the entire coherence group of projects as performed in *RTM*. A recent study by Kocaguneli et al. [43] concluded that an estimation that relies on a smaller number of more relevant analogues will result in better estimation performance than relying on a larger number of less relevant analogues. Our results are in agreement with Kocaguneli et al. as *LSA-X* achieved significantly improved estimation performance. This finding thus allows us to suggest interesting future studies to consider exploitation of the degree of relevance between new cases and their analogue as well as to further exploit productivity.

### 6.2 Findings

In this section, we further discuss ways to compensate for the limitation of *LSA-X* when a given dataset has $r(Pr, PrFactor)$ less than 0.30. This summarizes our findings based on all tables and results in this paper.

**Finding 1:** Table 6 shows the best feature subset selection method is specific for both solution adaptation techniques and datasets. Hence, for a particular set of data, identifying an appropriate feature subset selection method is just as important as determining the most suitable adaptation technique.

**Finding 2:** According to Table 5 and Table 6, for datasets with $r(Pr, PrFactor)$ less than 0.30, the *MSA* technique performed the best. Table 4 showed that *MSA* performed as well as *LSA-X* for some specific datasets with $r(Pr, PrFactor)$ greater than 0.30. Without any extension based on the *PrFactor*, in general, *MSA* would have provided the best accuracy. This finding shows that in software effort estimation, exploitation of $r(Pr, PrFactor)$ contributed to accuracy.

**Finding 3:** Results from all tables show that the solution adaptation techniques utilizing linear adjustment functions, e.g. *LSA-X*, are consistently more accurate than those using non-linear functions e.g. *NNet*. The results also show that and adaptation techniques that exploit productivity are more accurate than any other techniques. This finding supports the usefulness of exploiting productivity in software development effort estimation.

### 6.3 Threats to Validity

**Internal validity** questions whether different conclusions can be drawn with respect to the different parameters used in the experiments. The main possible question of internal validity is regarding the choice of the optimization parameters configured for each solution adaptation technique and for the baseline ABE framework. In our experiments, the optimization parameters that were configured for each solution adaptation technique and for the baseline ABE framework were selected to be as intuitive as possible. Whenever such an approach was available, we selected a more robust and intuitive configuration approach such as Baker's best k [38]. In cases where such general accepted procedure was not available, we chose an approach that commonly appeared in the literature, such as the approach we used to normalize all the continuous variables in the interval between 0 and 1. In cases where neither a general accepted nor a commonly used configuration was available, such as the criteria to select the coherence groups for the *RTM* technique, we adopted our own criteria that were sufficiently robust without any attempts to improve the existing techniques, since our main aim is not to find the best configuration parameter for any existing techniques.

**External validity** questions whether the results can be generalized. In this study, we used a wide range of datasets consisting of 1,188 software projects [18]. The selected datasets are varied in both size and characteristics, as they came from different organizations and were developed in different periods of time. The diversity of the datasets used this our study is the main reason that the *PrFactors* selected in different datasets are diversified. Compared to another recent assessment of solution adaption techniques [11] in which only 7 datasets were used, we believe that both the number and diversity of datasets used in this study offers a higher degree of validity than other studies of solution adaptation techniques, and is sufficient to justify generalizing our results.

In this study, we used 4 feature selection methods to avoid a possible bias because a particular feature subset selection method may perform well with some specific adaptation techniques. One possible issue of external validity regarding feature subset selection concerns the **Sfs** method. The improvement criteria of the method we used was the same as used in the study by Keung et al. [4] that was not based on ABE. The reason for this approach is that Keung et al. reported that a training model, e.g. the simple linear regression, used for this objective function has no influence on the accuracy of the same estimation model when it was used as an effort estimator. Thus, it is valid to use any model as the evaluation criteria for **Sfs** as long as the same criteria is applied across all experiments.

**Construct validity** questions whether or not we are measuring what we intend to measure. Software effort estimation studies are often criticized for inadequate use of evaluation metrics such as error measures. For example, use of *MMRE* by itself is commonly seen in the literature [26], despite widespread criticism of *MMRE* as an inappropriate and biased measure. However, in this study, we maximized the construct validity by using 5 robust error measures, all of which were guaranteed there robustness by a statistical method [26]. Furthermore, the statistical test method we applied to evaluate the performance of the adaptation techniques is a very robust variant of the Wilcoxon rank-sum test [36]. We believe that effort estimation using these measures as well as having the results tested by a robust statistical method offers a high degree of construct validity for this study.

## 7. Conclusion

This paper explores the possibility of exploiting productivity (*Pr*) and its influential factor (*PrFactor*) in the solution adaptation stage of analogy-based software development effort estimation. Productivity is one of the essential factors to analyze to estimate software development effort, and it is widely studied in many area of expertise [41], [42]. Our results show strong evidence that the proposed *LSA-X* adaptation technique, which exploits both *Pr* and *PrFactor* in its adaptation procedure, is a successful technique to generate a robust effort estimate. In an evaluation subject to a robust statistical test method using Wilcoxon rank-sum test (95% confidence), the proposed *LSA-X* technique outperformed 8 techniques commonly adopted in the literature and in practice [11], for datasets that exhibit a high correlation ($r \geq 0.30$) between *Pr* and *PrFactor*. For the other less common cases, our results showed that *LSA-X* has limited accuracy, and we suggest using another adaptation technique that also based on software size and productivity such as the *MSA* and the *RTM* techniques to compensate for this limitation.

Specifically, our detailed experiment evaluated both adaptation techniques and feature selection methods. The results suggested that:

- For a dataset with $r(Pr, PrFactor) \geq 0.30$, the most accurate results are obtained by adjusting the using the *LSA-X* technique and using all features (i.e. no feature subset selection is performed).
- For a dataset with $r(Pr, PrFactor) < 0.30$, it is more accurate to perform feature subset selection using the stepwise regression method and then adjust the effort using the *MSA* technique.

To date, there have been very few studies in software development effort estimation that thoroughly examined the influence of productivity on the improved estimation performance. For this study, the main futures work include a broader study of productivity and its influential factor since our findings have strongly supported the usefulness of exploiting them. Furthermore, since we can specify which datasets have accurate adaptation techniques and which have on-average inaccurate techniques, building an ensemble of techniques is an interesting future work. A recent study by Kocaguneli [13] suggests that combining multiple model-based software effort estimation methods will produce a consistently more accurate performance than any solo methods. One of the essential criteria for selecting the multiple methods is to choose only accurate estimation methods as resources to build an ensemble. Hence, our results offers a very good promise for future improvements of analogy-based estimation by exploring an ensemble of models built with accurate solution adaptation techniques, such as *LSA-X*, *MSA*, and *RTM*.

## Acknowledgements

This research was supported by JSPS KAKENHI Grant number 26330086, was conducted as a part of the Program for Advancing Strategic International Networks to Accelerate the Circulation of Talented Researchers, and was supported in part by the City University of Hong Kong research fund (Project number 7200354 and 7004222).

### References

[1] M. Shepperd and C. Schofield, "Estimating software project effort using analogies," IEEE Trans. Softw. Eng., vol.23, no.11, pp.736–743, 1997.

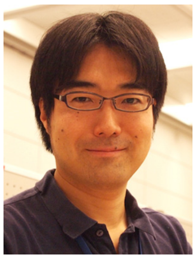
[2] J. Keung, "Software development cost estimation using analogy: A review," Proc. of the 2009 Australian Software Eng Conference, pp.327–336, 2009.

[3] C. Mair and M. Shepperd, "The consistency of empirical comparisons of regression and analogy-based software project cost prediction," Proc. of the 2005 International Symposium on Empirical Software Eng, pp.509–518, 2005.

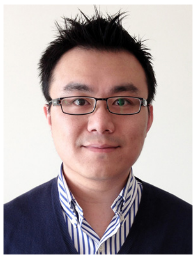
[4] J. Keung, E. Kocaguneli, and T. Menzies, "Finding conclusion stability for selecting the best effort predictor in software effort estimation," Automated Software Engineering, vol.20, no.4, pp.543–567, 2013.

[5] F. Walkerden and R. Jeffery, "An empirical study of analogy-based software effort estimation," Empir. Softw. Eng., vol.4, no.2, pp.135–158, 1999.
[6] C. Kirsopp, E. Mendes, R. Premraj, and M. Shepperd, "An empirical analysis of linear adaptation techniques for case-based prediction," Proc. 5th International Conference on Case-Based Reasoning: Research and Development, pp.231–245, 2003.
[7] M. Jørgensen, U. Indahl, and D. Sjøberg, "Software effort estimation by analogy and "regression toward the mean"," J. Syst. Softw., vol.68, no.3, pp.253–262, 2003.
[8] J. Li, G. Ruhe, A. Al-Emran, and M.M. Richter, "A flexible method for software effort estimation by analogy," Empir. Softw. Eng., vol.12, no.1, pp.65–106, 2007.
[9] N.-H. Chiu and S.-J. Huang, "The adjusted analogy-based software effort estimation based on similarity distances," J. Syst. Softw., vol.80, no.4, pp.628–640, 2007.
[10] Y.F. Li, M. Xie, and T.N. Goh, "A study of the non-linear adjustment for analogy based software cost estimation," Empir. Softw. Eng., vol.14, no.6, pp.603–643, 2009.
[11] M. Azzeh, "A replicated assessment and comparison of adaptation techniques for analogy-based effort estimation," Empir. Softw. Eng., vol.17, no.1-2, pp.90–127, 2012.
[12] B.W. Boehm, Software Engineering Economics, 1st ed., Prentice Hall PTR, Upper Saddle River, NJ, USA, 1981.
[13] E. Kocaguneli, T. Menzies, and J.W. Keung, "On the value of ensemble effort estimation," IEEE Trans. Softw. Eng., vol.38, no.6, pp.1403–1416, 2012.
[14] A. Aamodt and E. Plaza, "Case-based reasoning: foundational issues, methodological variations, and system approaches," AI Commun, vol.7, no.1, pp.39–59, 1994.
[15] M. Shepperd and G. Kadoda, "Comparing software prediction techniques using simulation," IEEE Trans. Softw. Eng., vol.27, no.11, pp.1014–1022, 2001.
[16] E. Mendes, N. Mosley, and S. Counsell, "A replicated assessment of the use of adaptation rules to improve web cost estimation," Proc. of the 2003 International Symposium on Empirical Software Eng, pp.100–109, 2003.
[17] B. Kitchenham and E. Mendes, "Software productivity measurement using multiple size measures," IEEE Trans. Softw. Eng., vol.30, no.12, pp.1023–1035, 2004.
[18] T. Menzies, M. Rees-Jones, R. Krishna, and C. Pape, "tera-promise: one of the largest repositories of se research data." http://openscience.us/repo/index.html, June 2015.
[19] C.F. Kemerer, "An empirical validation of software cost estimation models," Commun. ACM, vol.30, no.5, pp.416–429, 1987.
[20] Y. Miyazaki, M. Terakado, K. Ozaki, and H. Nozaki, "Robust regression for developing software estimation models," J. Syst. Softw., vol.27, no.1, pp.3–16, 1994.
[21] B. Kitchenham and K. Känsälä, "Inter-item correlations among function points," Proc. of the 15th International Conference on Software Eng, pp.477–480, IEEE, 1993.
[22] K. Maxwell, Applied Statistics for Software Managers, Prentice-Hall, Englewood Cliffs, NJ. 2002.
[23] A.J. Albrecht and J.E. Gaffney, "Software function, source lines of code, and development effort prediction: A software science validation," IEEE Trans. Softw. Eng., vol.9, no.6, pp.639–648, 1983.
[24] T. Menzies, D. Port, Z. Chen, J. Hihn, and S. Stukes, "Validation methods for calibrating software effort models," Proc. of the 27th international conference on Software engineering, pp.587–595, 2005.
[25] A. Bakır, B. Turhan, and A.B. Bener, "A new perspective on data homogeneity in software cost estimation: a study in the embedded systems domain," Softw. Qual. J., vol.18, no.1, pp.57–80, 2010.
[26] T. Foss, E. Stensrud, B. Kitchenham, and I. Myrtveit, "A simulation study of the model evaluation criterion mmre," IEEE Trans. Softw. Eng., vol.29, no.11, pp.985–995, 2003.
[27] P. Phannachitta, A. Monden, J. Keung, and K. Matsumoto, "Case consistency: a necessary data quality property for software engineering data sets," Proc. of the 19th International Conference on Evaluation and Assessment in Software Eng, p.19, ACM, 2015.
[28] Z. Chen, T. Menzies, D. Port, and B. Boehm, "Feature subset selection can improve software cost estimation accuracy," ACM SIGSOFT Software Engineering Notes, vol.30, no.4, pp.1–6, 2005.
[29] E. Alpaydin, Introduction to machine learning, MIT Press, 2014.
[30] E. Kocaguneli, T. Menzies, J. Hihn, and B.H. Kang, "Size doesn't matter?: On the value of software size features for effort estimation," Proc. of the 8th International Conference on Predictive Models in Software Eng, New York, NY, USA, pp.89–98, ACM, 2012.
[31] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," Proc. 14th International Joint Conference on Artificial Intelligence, pp.1137–1143, 1995.
[32] E. Kocaguneli and T. Menzies, "Software effort models should be assessed via leave-one-out validation," J. Syst. Softw., vol.86, no.7, pp.1879–1890, 2013.
[33] M.V. Kosti, N. Mittas, and L. Angelis, "Alternative methods using similarities in software effort estimation," Proc. of the 8th International Conference on Predictive Models in Software Eng, pp.59–68, 2012.
[34] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," J. Mach. Learn. Res., vol.7, pp.1–30, 2006.
[35] E. Brunner, U. Munzel, and M.L. Puri, "The multivariate nonparametric Behrens–Fisher problem," J. Stat. Plan. Infer., vol.108, no.1, pp.37–53, 2002.
[36] D.W. Zimmerman, "Statistical significance levels of nonparametric tests biased by heterogeneous variances of treatment groups," The Journal of General Psychology, vol.127, no.4, pp.354–364, 2000.
[37] E. Kocaguneli, T. Menzies, J. Keung, D. Cok, and R. Madachy, "Active learning and effort estimation: Finding the essential content of software effort estimation data," IEEE Trans. Softw. Eng., vol.39, no.8, pp.1040–1053, 2013.
[38] D.R. Baker, "A hybrid approach to expert and model based effort estimation," Master's thesis, Lane Department of Computer Science and Electrical Engineering, West Virginia University, 2007.
[39] L.C. Briand and I. Wieczorek, "Resource estimation in software engineering," Encyclopedia of software engineering, 2002.
[40] E. Kocaguneli, T. Menzies, and J.W. Keung, "Kernel methods for software effort estimation - effects of different kernel functions and bandwidths on estimation accuracy," Empir. Softw. Eng., vol.18, no.1, pp.1–24, 2013.
[41] G.S. de Aquino and S.R. De Lemos Meira, "An Approach to Measure Value-Based Productivity in Software Projects," 2009 Ninth International Conference on Quality Software, pp.383–389, 2009.
[42] J. Kijne, T. Tuong, J. Bennett, B. Bouman, T. Oweis, et al., "Ensuring food security via improvement in crop water productivity," Challenge Program on water and Food: Background Paper, vol.1, 2003.
[43] E. Kocaguneli, T. Menzies, A. Bener, and J.W. Keung, "Exploiting the essential assumptions of analogy-based effort estimation," IEEE Trans. Softw. Eng., vol.38, no.2, pp.425–438, 2012.

**Passakorn Phannachitta** received his BE (Hons) in Computer Engineering from Kasetsart University, Thailand. Now he is currently a Ph.D. student at Nara Institute of Science and Technology, Japan. His research interests include software development effort estimation, data mining technique, and high performance computing.



**Akito Monden** received the BE degree in 1994 in electrical engineering from Nagoya University, and the ME and DE degrees in 1996 and 1998 in information science from Nara Institute of Science and Technology. He is a professor in Graduate School of Natural Science and Technology at Okayama University, Japan. He was honorary research fellow at the University of Auckland, New Zealand (2003–2004). He is a member of the IEEE, ACM, IEICE, IPSJ and JSSST.



**Jacky Keung** received his B.Sc (Hons) in Computer Science from the University of Sydney, and his Ph.D in Software Engineering from the University of New South Wales, Australia. He is an active software engineering researcher, and his main research area is in software effort and cost estimation, empirical modeling and evaluation of complex systems, and intensive data mining for software engineering data in the cloud.

**Kenichi Matsumoto** received the B.E., M.E., and PhD degrees in Information and Computer sciences from Osaka University, Japan, in 1985, 1987, 1990, respectively. Dr. Matsumoto is currently a professor in the Graduate School of Information Science at Nara Institute Science and Technology, Japan. His research interests include software measurement and software process. He is a senior member of the IEEE, and a member of the ACM and IPSJ.