

Insightful analogy-based software development effort estimation through selective classification and localization

Vahid Khatibi Bardsiri · Elham Khatibi

Received: 30 December 2013 / Accepted: 24 November 2014 / Published online: 5 December 2014
© Springer-Verlag London 2014

Abstract Accurate development effort estimation is a challenging issue in the management of software projects because it can considerably affect the planning and scheduling of a software project. Over the past few years, many algorithmic and non-algorithmic methods have been proposed to estimate the development effort in the early stages of project. Due to simplicity and estimation capability, analogy-based estimation (ABE) method has been widely accepted by researchers in this area. In spite of the fact that ABE is an efficient estimation method, it suffers from the non-normality and heterogeneous nature of software project datasets. Although prior studies have strived to remedy this issue by weighting, soft computing, and clustering techniques, the estimate accuracy is still not convincing and attempts are ongoing to reach more reliable estimates. The problem is that prior ABE-based studies have not considered the nature of software projects in the estimation process. This paper aims to show the effect of selective project classification and estimation process localization on the performance of ABE. An exhaustive investigation is conducted based on different development types, organization types, and development platforms as three underlying attributes in software projects. An evaluation framework is designed to reveal the ABE performance when it is combined with the proposed classification. A real dataset that includes 448 software projects is utilized for the evaluation purposes. The promising results showed that the estimate accuracy is significantly improved and the estimation process is considerably expedited if the nature of software projects is considered in the ABE method.

Keywords Software projects · Selective classification · Effort estimation · Analogy method

1 Introduction

Accurate estimation of development effort has become a challenging issue in the management of software projects over the past few years [4,28]. As a matter of fact, the special characteristics of software projects make the process of estimating more difficult than it may seem. The demands and requirements of software projects are naturally unstable, software development frameworks and related hardware platforms are changing continuously and the progress of project is invisible due to logical work. Moreover, unlike other types of projects (building construction, material production, etc.), developers are confronted with an intangible product whose specifications may not be completely obvious at the early stages of the project.

Due to the above-mentioned problems, numerous studies have been conducted to find reliable methods to estimate software development effort at the early stages of software projects. Researchers are confronted with the major problem of uncertain information. Since the available information is insufficient for use in equations, relations, formulas, etc., most previous research works in this area have found that comparison-based estimation is the most adaptable technique to use for the purposes of software development effort estimation. The selection of previous completed projects similar in nature to the target project and estimation of effort based on the effort reported for the selected projects is a widely accepted procedure. This method is called as analogy-based estimation (ABE) [39].

ABE is a simple and capable model in the field of software development effort estimation. However, the complex-

V. Khatibi Bardsiri (✉) · E. Khatibi
Department of Computer Engineering,
Bardsir Branch, Islamic Azad University, Kerman, Iran
e-mail: kvahid2@live.utm.my

ity and non-normality of project attributes decrease the estimate accuracy achieved by this model. Although more than ten years have passed since the invention of ABE, researchers are still interested in this model and continue to strive to improve its performance. In fact, ABE is at center of studies done in the area of software development effort estimation.

A wide range of statistical techniques and soft computing methods has been employed in an attempt to increase the performance of ABE model. Despite improvements in performance, estimate accuracy is still not convincing and there is no strong agreement as to which individual effort estimation model is the best [19,23]. Consequently, numerous papers are published in this area every year. The important issue is that prior studies have not considered the importance of software projects in the estimation process. In other words, prior studies have considered software projects as similar to other types of data and consequently employed techniques similar to those used to solve other problems.

A comparison between two software projects that are different in underlying attributes is a common mistake frequently seen in the prior studies. In fact, this type of comparisons simply leads to based estimates. Recently, researchers have found that the problem of software development effort estimation needs to be solved through classification of projects and localization of estimation process [7,21,32,34]. However, these studies have utilized classical methods for classification purpose regardless of the nature of projects. Furthermore, prior studies have relied on regression-based modes to conduct the localization process.

To consider the nature of software projects in the estimation process, this paper aims to propose selective classification instead of classical methods. Moreover, the process of localization is conducted by ABE which is a widely accepted estimation model. A combination of ABE and the selective classification can reveal the lack of attention paid to the different nature of software projects in prior comparison-based estimation models, which is the main objective of this paper.

The rest of the current paper is organized into eight sections as follows: Sect. 2 includes the principles of analogy-based estimation method. Section 3 explains the calculation of function point. The related work is presented in Sect. 4 and the proposed framework is explained in Sect. 5. The experimental results are explained in Sect. 6. Section 7 illustrates efficiency analysis and Sect. 8 includes threats to validity. Finally, the conclusion and future work are presented in Sect. 9.

2 Analogy-based estimation (ABE)

ABE model was proposed by [39] to be used instead of algorithmic models. In this model, the development effort is estimated by a comparison process in which the projects similar

to a new project are selected. The selected projects effort is then utilized to estimate the effort of new project. Due to simplicity and estimation capability, ABE has been extensively used in terms of software development effort estimation. Basically, ABE includes four components:

- (i) historical dataset
- (ii) similarity function
- (iii) the associated retrieval rules
- (iv) solution function

The ABE estimation process is carried out according to the following steps:

1. Gathering previous projects information and producing a historical dataset
2. Choosing attributes such as FP and LOC for comparison purpose
3. Retrieving previous projects similar to new project
4. Estimating the effort of new project

2.1 Similarity function

ABE uses a similarity function that compares the attributes of two projects to determine the level of similarity. There are two popular similarity functions, Euclidean similarity (ES) and Manhattan similarity (MS) [39]. Equation 1 shows the Euclidean similarity function.

$$\text{Sim}(p, p') = \frac{1}{\left[\sqrt{\sum_{i=1}^n w_i \text{Dis}(f_i, f'_i)} + \delta \right]} \quad \delta = 0.0001$$

$$\text{Dis}(f_i, f'_i) = \begin{cases} (f_i - f'_i)^2 & \text{if } f_i \text{ and } f'_i \text{ are numerical or ordinal} \\ 0 & \text{if } f_i \text{ and } f'_i \text{ are nominal and } f_i = f'_i \\ 1 & \text{if } f_i \text{ and } f'_i \text{ are nominal and } f_i \neq f'_i \end{cases} \quad (1)$$

Where p and p' are the projects to be compared while w_i is the weight assigned to each attribute. The weight can vary between 0 and 1. Moreover, f_i and f'_i display the i th attribute of each project and n demonstrates the number of attributes. δ is utilized for obtaining a non-zero result. The MS formula is very similar to that of ES but it computes the absolute difference between the attributes. Equation 2 shows the Manhattan similarity function.

$$\text{Sim}(p, p') = \frac{1}{\left[\sum_{i=1}^n w_i \text{Dis}(f_i, f'_i) + \delta \right]} \quad \delta = 0.0001$$

$$\text{Dis}(f_i, f'_i) = \begin{cases} |f_i - f'_i| & \text{if } f_i \text{ and } f'_i \text{ are numerical or ordinal} \\ 0 & \text{if } f_i \text{ and } f'_i \text{ are nominal and } f_i = f'_i \\ 1 & \text{if } f_i \text{ and } f'_i \text{ are nominal and } f_i \neq f'_i \end{cases} \quad (2)$$

In addition, there are other similarity functions such as rank mean similarity [41], maximum distance similarity and Minkowski similarity [2]. In some studies, several types of similarity functions have been employed to determine the best performance of ABE because there is no strong agreement as to which similarity function is the best [8,20,25–28,39].

2.2 Solution function

Solution function is utilized to estimate the software development effort by considering the similar projects found by the similarity function. The popular solution functions are: closest analogy as the most similar project [42], average of most similar projects [39], median of most similar projects [2] and inverse distance weighted mean [18]. The average describes the average value of effort obtained from K most similar projects, where $K > 1$. The median describes the median value of effort obtained from K most similar projects, where $K > 2$. The inverse distance weighted mean adjusts the portion of each project in k th estimation by means of Eq. 3.

$$C_p = \sum_{k=1}^K \frac{\text{Sim}(p, p_k)}{\sum_{i=1}^K \text{Sim}(p, p_i)} C_{p_k} \tag{3}$$

Where p shows the new project, p_k illustrates the k th most similar project, C_{p_k} is the effort value for the k th most similar project (p_k), $\text{Sim}(p, p_k)$ is the similarity between projects p_k and p while K is the total number of most similar projects. Different types of solution functions have been employed in the previous studies. Several studies used only one solution function [14,17,26,42] while some other studies used several types of solution function [2,28,31].

3 Function point size estimates

Function point (FP) metric was introduced by [1] to measure the functionality of a project. FP estimates are made by determining the indicators of user inputs, user outputs, logical files, inquiries and interfaces. A complexity degree, varies between 1 and 3, is defined to assign to each indicator. 1, 2 and 3 stand for simple, medium and complex degree, respectively. In addition, it is necessary to define a weight for each indicator, which can be determined between 3 and 15. In the first stage, the number of indicator should be counted and it must then be multiplied by the related weight. In general, unadjusted function point (UFP) computed through Eq. 4.

$$\text{UFP} = \sum_{i=1}^5 \sum_{j=1}^3 N_{ij} W_{ij} \tag{4}$$

Where N_{ij} is the number of indicator i with complexity j while W_{ij} is the weight of indicator i having complexity j . According to the previous experiences, the function point

Table 1 Function point components

F1	Reliable back-up and recovery	F8	Data communications
F2	Distributed functions	F9	Performance
F3	Heavily used configuration	F10	Online data entry
F4	Operational ease	F11	Online update
F5	Complex interface	F12	Complex processing
F6	Reusability	F13	Installation ease
F7	Multiple sites	F14	Facilitate change

is useful for estimation in software projects because it can be computed based on the requirement specifications at the early stages of a project. To compute FP, UFP should be multiplied by a technical complexity factor (TCF) which is obtained from the components described in Table 1.

A component value varies in a 0–5 range in which 0 indicates that the associated component has no effect on the project while 5 means that the component is very important. The range of TCF is between 0.65 (if all F_i are 0) and 1.35 (if all F_i are 5). Equations 5 and 6 show TCF and FP computation procedure, respectively.

$$\text{TCF} = 0.65 + 0.01(\text{SUM}(F_i)) \tag{5}$$

$$\text{FP} = \text{UFP} \times \text{TCF} \tag{6}$$

4 Related work

Due to simplicity and flexibility, ABE has been frequently used in the hybrid effort estimation models proposed over the past few years. In the ABE method, the development effort of a new project is estimated through a comparison between the new project and those completed in the past. Similarity and solution functions are the main components of ABE to measure the similarity level between two projects and to estimate the development effort using similar projects, respectively. The combinations of ABE and the genetic algorithm [8,14,29,33], ABE and PSO [22,43], ABE and ANN [21,28], ABE and grey [5,13,40], ABE and outlier elimination techniques [38], ABE and principle component analysis (PCA) [16], ABE and regression [35] as well as ABE and rough set theory [26] are some instances of hybrid models constructed using ABE.

The complexity and non-normality of software project attributes have been frequently elaborated in prior studies and the generalization of estimation models has been extensively mentioned as a critical issue. Dolado [12] reported that mathematical-based effort estimation models are unable to be an universal model. It was the first alert for researchers to invent more flexible and reliable estimation models.

Finally, segmented software development effort estimation models were introduced to remedy the problem of inconsistency in software projects [10,37]. The authors proposed a

model in which the projects are divided into several clusters using a clustering algorithm (EM and M5). For each cluster a regression-based equation was computed as the estimation model assigned to that cluster. The empirical study of segmented models showed that they considerably increased the accuracy of estimates compared to models constructed on whole data. The authors investigated the proposed segmented models on a large number of datasets to show the estimate capabilities in a wider space [11].

Although the segmented estimation models are a novel and helpful idea in the area of software development effort estimation, there are several serious drawbacks that restrict the application and adaptation of these models. The proposed segmented models have been constructed based on mathematical equations that are strongly not recommended for software effort estimation due to low level of flexibility and adaptability. Relying on regression-based estimation models is quite risky because these models may fail to reach accurate estimates in the clusters having complex and non-linear relationships between attributes. Furthermore, the project clustering has been performed regardless of the software project type, which can have a negative effect on the accuracy of estimates.

Recently, [32] have conducted an empirical study to investigate the performance of the global effort estimation models on the local areas. The authors classified the projects into several groups and investigated the performance of a global model on the obtained groups. The results of this study showed that a global model may be not a suitable model to be used in the local areas despite showing a good performance in the global area.

In another study, the effect of localization was investigated through the regression-based effort and defect estimation models [7]. The proposed model was constructed in two separate steps: using whole projects and classified projects. The results showed that the accuracy of model was substantially increased through the local models. Moreover, [34,36] have reported that localization of estimation process can improve the accuracy of estimates.

Although researchers have found that the localization plays a crucial role in the software development effort estimation, the localization domain has restricted to regression and statistical-based models. Moreover, the nature of software projects has not been considered in the localization process because blind and classical classification methods have been employed to classify the projects.

It must be noted that prior studies have not considered the underlying differences of software projects in the ABE-based models. In other words, the effect of localization on the performance of ABE-based models has not been investigated yet. The problem is that soft computing techniques are unable to process the software projects unless the underlying attributes are considered in the estimation process.

The evaluation framework proposed in this paper combines the selective classification of software projects and the ABE model to show the effect of attention to the nature of projects, which is a real gap in the area of software development effort estimation.

5 Proposed framework

5.1 Selective classification

Inconsistency and non-normality are two important specifications of software project attributes, which make the effort estimation difficult and complicated. Therefore, it is seemingly impossible to propose an overall estimation model in which all types of software projects can be estimated. In particular, comparison-based models like ABE are strongly dependent on the quality of data used for the purpose of estimating. The accuracy of ABE is considerably decreased when it is utilized on datasets that include heterogeneous and inconsistent projects.

As stated before, ABE compares the projects attributes to determine the similarity level. This is the main component of the estimation process in ABE, which can be significantly affected by the non-normality of project attributes in historical datasets. The classification of projects into several groups has attracted the attention of researchers to alleviate the problem of inconsistency in this field. As a matter of fact, the classification of projects decreases the number of comparisons performed in ABE because the historical dataset is decreased to several smaller sets. On the other hand, it leads to more smooth and consistent sets of projects. Therefore, it is expected that more reliable comparisons and more accurate estimates are achieved through classification process.

Prior studies have utilized classical classification methods such as *c*-means [3,21], *k*-means [30], M5 [11], and GAC [23] to localize the estimation process. There are several drawbacks that restrict the application of these methods in the field of software development effort estimation. First of all, blind classification of software projects may produce a cluster whose projects are different in underlying attributes and are similar in worthless attributes. This is because the classical methods rely on similarity of project attributes regardless of the nature and importance of attributes. Therefore, the smoothness and consistency of projects located in the clusters cannot be assured using the classical classification methods.

Moreover, the clusters produced by the classical methods are meaningless and cannot be interpreted because they are comprised of different types of software projects. The number of clusters is also a challenging issue due to lack of a specified method to determine the best number of clusters.

The problems related to the classical classification methods support the idea of the selective classification of software projects employed by this paper. The selective classification is performed based on the underlying project attributes. The underlying attributes refer to those attributes that are very important in the software development process and have a considerable effect on the development effort. An underlying attribute is a categorical attribute which can be determined in the early stages of a software project.

The numerical attributes cannot be utilized for the classification purpose. In addition, the categorical attributes such as architecture, programming language, and standard should not be considered as the underlying attributes because they cannot be determined in the early stages of a software project. A comprehensive evaluation performed by ISBSG company reviewers [15] on 5,052 software projects (from all over the world) shows that development type, organization type and development platform considerably affect the development effort of software projects. These three attributes are determinable in the early stage of any software projects. Moreover, a large number of existing software project data sets includes these attributes, which is suitable to generalize the application of the selective classification. Furthermore, recent studies related to the software development effort estimation have frequently used the mentioned attributes to construct the estimation models [6, 24, 28]. Finally, these attributes are the fundamental attributes in software development process, from software engineering perspective. Therefore, development type, organization type, and development platform are considered to conduct the selective classification.

5.2 Evaluation procedure

To clarify the effect of selective project classification, the performance of ABE must be evaluated in two different steps. In the first step, the ABE performance is evaluated using whole historical projects while in the second it is evaluated using the classified projects. All the projects are classified into several sets based on the selected underlying attributes, as seen in Fig. 1. The performance of ABE is then evaluated using all classified sets and whole dataset. In the proposed framework, leave-one-out technique is utilized for evaluation purpose.

For each set (D), a project (p_i) is selected that has not been visited so far. P_i is considered as a new project to be estimated while the other projects are considered as the historical dataset (HD) used for comparison purpose. Therefore, the historical dataset and the new project are applied to ABE where the development effort is estimated. The estimated effort and actual effort of the new project are recorded for further investigation. This process is repeated for all the projects that exist in the set. Once the estimation is completed for all the projects, the performance metrics are computed for

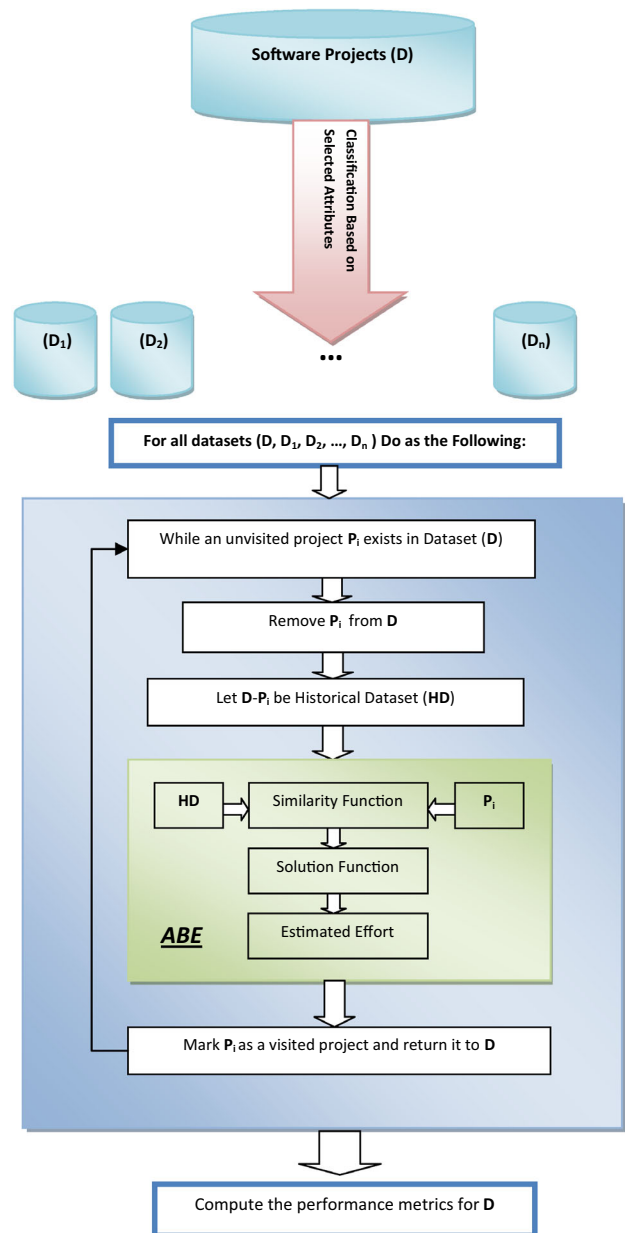


Fig. 1 The proposed framework

the current set. The next set is then selected and the above-mentioned steps are performed, accordingly.

This process is continued until the performance metrics are computed for all the classified sets. Therefore, at the end of this process, the performance metrics related to the classified sets and whole dataset are available to be compared.

6 Experimental design

6.1 Data preparation

International software benchmarking standard group (ISBSG) is a company located in Australia. It collects

Table 2 Description of ISBSG dataset

Attribute	Description	Min	Max	Mean	Median	Std Dev
InpCont	Input count	3	2,221	152.20	72	226.96
OutCont	Output count	4	2,455	141.25	65.50	210.06
EnqCont	Enquiry count	3	1,306	115.81	64.50	155.39
FileCont	File count	7	1,732	130.82	68.50	184.11
IntCont	Interface count	5	1,572	70.87	30	147.63
AFP	Adjusted function point	29	7,633	625.66	380	770.13
DevType	Development type	1	3	–	–	–
OrgType	Organization type	1	6	–	–	–
DevPlat	Development platform	1	4	–	–	–
NorEffort (h)	Normalized effort	64	6,0826	5,588.65	3,216	7,095.64

the information related to software projects from all over the world. In this paper, ISBSG dataset release 11 [15] is employed. It contains the detailed information about 5,052 software projects. A filtration process must be conducted to select an appropriate and reliable subset of ISBSG projects.

In this study, the quality of data is very important because the proposed models are constructed through an intensive data analysis. Therefore, poor data validation may lead to unreliable results and misleading conclusions. Fortunately, validation process in selected data repository can be easily performed. ISBSG group has defined four quality levels for published data, which are described as follows:

A = The data submitted was assessed as being sound with nothing being identified that might affect its integrity.

B = The submission appears fundamentally sound but there are some factors which could affect the integrity of the submitted data.

C = Due to significant data not being provided, it was not possible to assess the integrity of the submitted data.

D = Due to one factor or a combination of factors, little credibility should be given to the submitted data.

Since this research was conducted based on in-depth analysis of data, the most significant data must be selected. Therefore, the software projects that have quality levels of A or B were selected in this step. Since there is a doubt in integrity and validity of projects with quality levels of C or D, they were removed from the final set of projects.

For the projects whose total effort has not been reported, ISBSG estimates it in the form of normalized effort and calculates a variable called normalized ratio. The normalized ratio is obtained from the division of the normalized effort by the reported effort, which shows the difference between the reported and estimated effort. According to the recommendations by ISBSG, the projects that have a normalized ratio greater than 1.2 are considered risky and unreliable.

Therefore, all projects with normalized ratio greater than 1.2 were removed from the final set. Moreover, the projects with size measurement unit other than IFPUG were excluded. Finally, the projects having resource level other than {1, 2: development effort and support effort, respectively} were omitted. The statistical information related to ISBSG attributes is presented in Table 2. From the table, there are seven numerical and three nominal attributes in the selected subset of ISBSG. After removing the projects whose attribute values have not been reported, a total of 448 software projects are achieved as the final set.

6.2 Performance metrics

In this paper, performance evaluation is performed using a number of widely accepted metrics including magnitude of relative error (MRE), mean magnitude of relative error (MMRE), and percentage of the prediction (PRED) [39], which are computed as follows.

$$RE = \frac{(\text{Estimate}-\text{Actual})}{\text{Actual}} \quad (7)$$

$$MRE = \frac{|\text{Estimated}-\text{Actual}|}{\text{Actual}} \quad (8)$$

$$MMRE = \frac{\sum_{i=1}^N MRE}{N} \quad (9)$$

$$PRED(X) = \frac{A}{N} \quad (10)$$

Where A is the number of projects with MRE less than or equal to X and N is the number of projects that exist in the testing set. The acceptable performance of a software development effort estimation model is defined based on $MRE \leq 0.25$ [9]. In other words, the value of 0.25 has been defined to separate the accurate and inaccurate estimates. The number of estimates having $MRE \leq 0.25$, as the most well-known performance metric, has been widely accepted by researchers in this field. For all the estimators, MMRE must be minimized while $PRED(0.25)$ must be maximized.

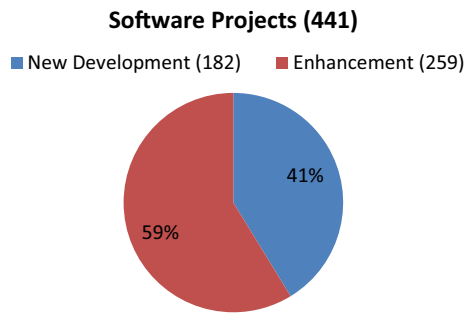


Fig. 2 Classification based on development type

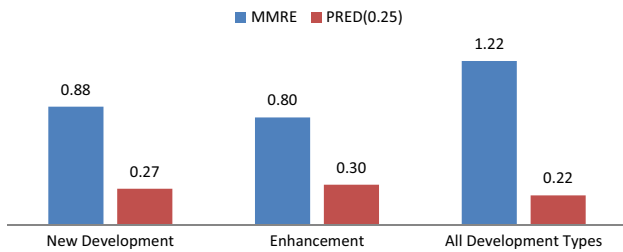


Fig. 3 Comparison of performance metrics after classification of projects

6.3 Classification based on development type

Development type is a key attribute used in the proposed framework to classify the projects. According to the attributes that exist in the data repository, the development type is divided into three main groups: new-development, enhancement and re-development. The term “New-development” refers to those projects defined to respond a new demand (no prior history for demand). “Enhancement” means that a project must be defined to increase and improve the capabilities of existing software. Finally, “re-development” refers to projects defined to improve the structure of existing software through new technologies and tools without applying any changes to software capabilities. This type of development is commonly used for maintenance purpose. In this study, the projects are classified based on new-development and enhancement because there is a few projects whose development type has been reported as re-development (in ISBSG dataset). Figure 2 depicts the distribution of projects based on development type. After exclusion of unreliable projects and those include missing attributes, a total of 441 projects are achieved (182 new-development projects and 259 enhancement projects). Figure 3 shows the results obtained from applying the selective classification to the dataset. Indeed, the performance of ABE has been evaluated on whole dataset and classified projects. As stated in the proposed framework, the comparison scope is reduced using classification.

New-development, enhancement and all development types have been treated as D1, D2 and D, respectively (vari-

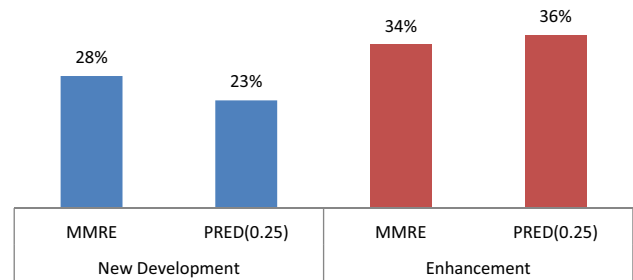


Fig. 4 Percentage of improvement achieved by classification

ables utilized in Fig. 1). Therefore, in this case there are three main sets that must be considered to evaluate the performance of ABE. As seen in Fig. 3, the performance metrics of MMRE and PRED(0.25) are computed for all three sets, separately. The performance of ABE is strongly dependent on the number of projects that exist in the historical dataset. Therefore, it may seem that ABE estimations on the whole dataset are more accurate than that of classified sets.

From Fig. 3, this hypothesis is rejected because accuracy of estimates on classified sets is more than that for whole dataset. The results achieved by the proposed framework implies that however classification reduces the number of projects in the historical dataset, it can be helpful to decrease the inconsistency and non-normality of projects and to increase the quality of comparisons.

Figure 4 depicts the percentage of improvement obtained from applying the selective classification to ABE. To calculate the percentage of improvement for MMRE and PRED, Eqs. 11 and 12 are utilized, respectively. As seen in the figure, both MMRE and PRED(0.25) are improved when the classification is utilized in the proposed framework. In terms of new-development projects, the percentage of improvement for MMRE and PRED(0.25) is 28 and 23 %, respectively.

$$\frac{MMRE_{Whole} - MMRE_{Classified}}{MMRE_{Whole}} \tag{11}$$

$$\frac{PRED_{Classified} - PRED_{Whole}}{PRED_{Whole}} \tag{12}$$

On the other hand, MMRE and PRED(0.25) have, respectively, been improved by 34 and 36 % for enhancement projects. The improvement of performance metrics for enhancement projects is greater than that for the new-development projects. This implies that the projects with enhancement development type are more consistent than another. In other words, the level of normality between the project attributes and effort in enhancement projects is greater than that for new-development projects (only valid for selected projects).

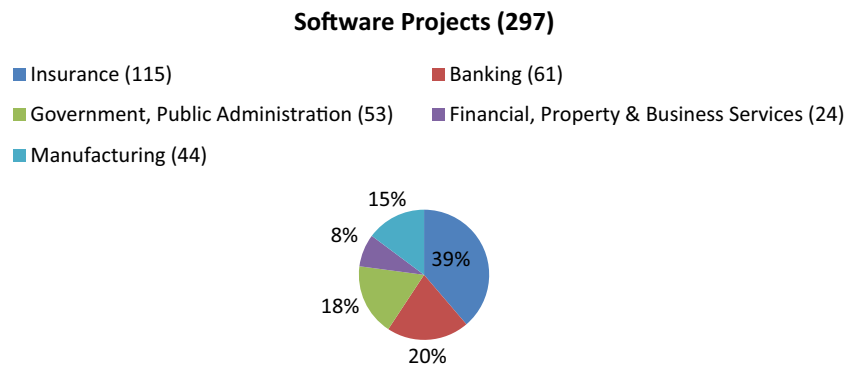


Fig. 5 Classification of projects based on organization type

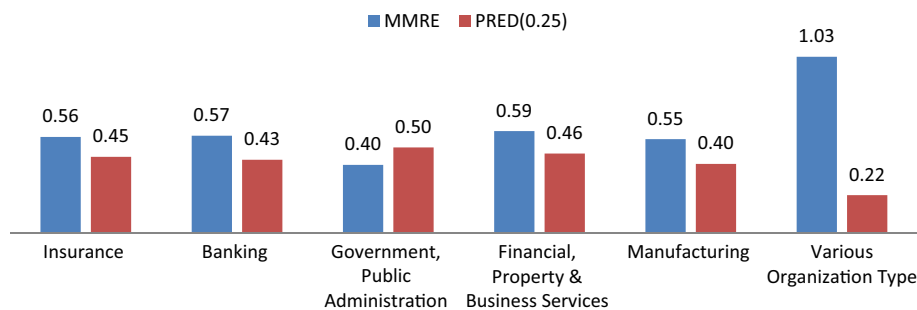


Fig. 6 Comparison of performance metrics after classification of projects

6.4 Classification based on organization type

Organization type is another attribute considered for projects classification purpose. This attribute specifies the organization where the software project is conducted for. The organization type can affect the effort required to develop the software because complexity and critical level of projects change from one organization type to another. Priorities, adaptability, demand types and limitations are the most important factors that may differ among different organization types. According to the data repository, organization type can be divided into five main groups: Insurance, Banking, Government, Financial and Manufacturing. After exclusion of unreliable projects and those that include unreported attributes, a total of 297 projects are achieved, as seen in Fig. 5. Insurance and financial are the largest and smallest sets with 115 and 24 projects, respectively.

Figure 6 depicts the results of projects classification based on the organization type. In this case, there are six sets treated as D1, D2, D3, D4, D5 and D in the proposed framework. First five sets are the classified projects while the last one is the whole dataset. As seen in the figure, the performance metrics of MMRE and PRED(0.25) are improved when the projects are classified based on the organization type.

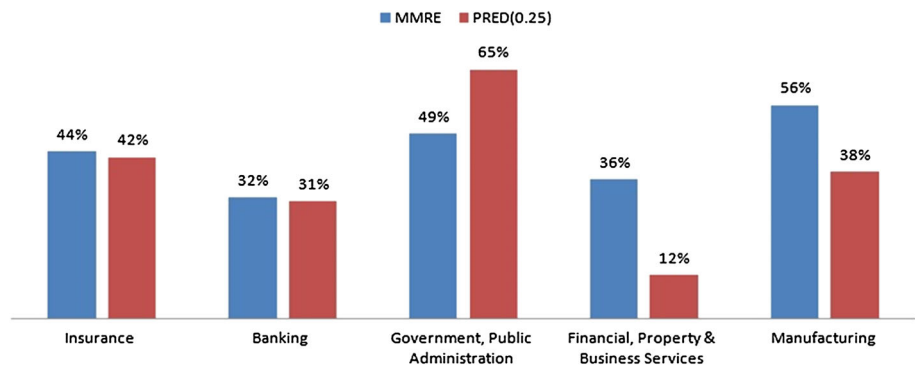
The reason behind improvement of performance metrics is that the classification of projects makes the historical dataset smooth and consistent for the purpose of comparison per-

formed in ABE. Therefore, high diversity of organization types is one of the problems ABE suffers from and the classification of projects based on the organization type can considerably alleviate this problem. It must be noted that if this type of classification can lead to improvement of performance metrics, in addition to increase of accuracy, the number of comparisons is substantially reduced. This implies that the speed of estimation process is increased in ABE. Therefore, the results obtained from classification of projects, based on the development type and organization type, confirm the decrease of comparisons and increase of accuracy.

Figure 7 displays the percentage of improvement achieved by classification of projects based on the organization type. In overall, it is observed that the value of MMRE has been improved more than 31 % whereas the value of PRED(0.25) has been improved more than 11 % in all sets. The average of improvement for MMRE and PRED(0.15) is 43 and 38 %, respectively. Regarding PRED(0.25), the highest and the lowest percentage of improvement are achieved by government and financial organization types, respectively. On the other hand, Manufacturing and Banking organization types reach the highest and lowest percentage of improvement for MMRE. In total, the best performance of ABE is related to the organization type of government.

Since the performance metrics have been improved in all different sets, it can be concluded that the selective classification of projects based on the organization type can be a

Fig. 7 Percentage of improvement achieved by classification



Software Projects(448)

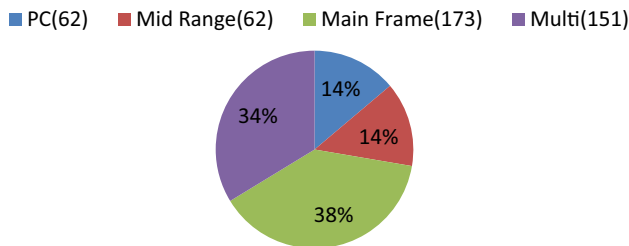


Fig. 8 Classification of projects based on development platform

solution to the problem of non-normality and inconsistency that make the process of estimation difficult in ABE. Compared to the classification of projects based on development type, the results obtained from organization type classification show more accurate estimates according to MMRE and PRED(0.25).

6.5 Classification based on development platform

One of the most important project attributes that considerably affect the development effort is development platform. This attribute includes the requirements such as distributed systems, parallel computing, multi user stations and other options related to application environment. As the platform requirements increase, the development effort is increased. Therefore, the comparison between the projects having different development platforms may lead to inaccurate estimates. According to ISBSG dataset, the development platform is divided into four main groups of PC, Mid-Range, Main Frame and Multi. Figure 8 displays the classification of software projects based on the development platform.

As seen in Fig. 8, a total of 448 software projects have been classified into four groups based on the development platform. The number of projects whose development platform has been reported Main Frame is 173, which includes the most percentage of projects (38 %). PC, Mid Range,

Main Frame, Multi and all projects (various development platforms) are considered as D_1, D_2, D_3, D_4 and D , respectively, in the proposed framework. The results achieved by the classification of projects based on development platform are depicted in Fig. 9.

According to Fig. 9, both MMRE and PRED(0.25) are improved when the classified sets are utilized by ABE. The greatest value of MMRE and the lowest value of PRED(0.25) are reported for various development platforms. This implies that lack of paying attention to development platform can considerably affect the quality of comparisons carried out in ABE. Indeed, comparison of performance metrics reveals the important role of development platform in development effort estimation.

Figure 10 shows the percentage of improvement achieved by the selective classification of projects based on development platform. The average of improvement for MMRE and PRED(0.25) is 37 and 40 %, respectively. It is observed that the lowest improvement of MMRE and the highest improvement of PRED(0.25) are achieved by the development platform of Mid Range. The minimum improvement is 31 and 17 % for MMRE and PRED(0.25), which approves that the classification of projects based on development platform can significantly affect the accuracy of estimates achieved by ABE.

The reason behind improvement of performance metrics is that the classification of projects makes the historical dataset smooth and consistent for the purpose of efficient comparison in ABE. Therefore, the high diversity of software projects is one of the most important obstacles ABE suffers from. The classification of projects based on the key attributes can considerably alleviate this problem. As stated before, besides increase of accuracy, the selective classification leads to reduce the number of comparisons because the size of historical dataset is decreased. This implies that the speed of estimation process is also increased in ABE. Therefore, the results obtained from the classification of projects, based on the development type, organization type and development platform, confirm the decrease of comparisons and increase of accuracy.

Fig. 9 Comparison of performance metrics after classification of projects

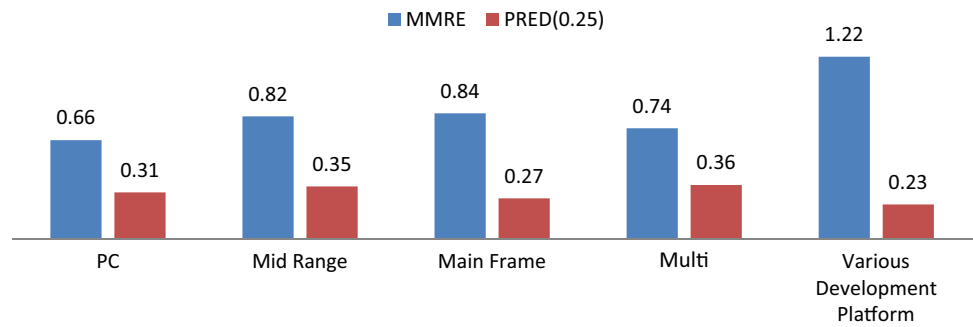
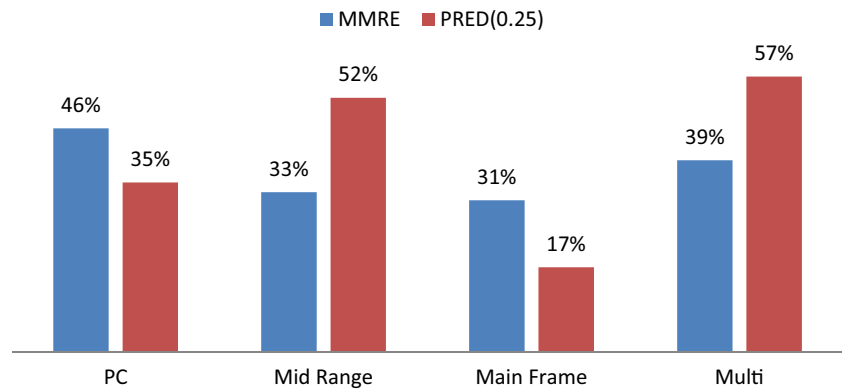


Fig. 10 Percentage of improvement achieved by classification



7 Efficiency analysis

The main part of analogy-based estimation is the comparisons performed in the similarity function. As the number of projects and the number of attributes are increased, the number of comparisons is also considerably increased. Indeed, the high number of comparisons is a drawback of ABE because it leads to increase of estimation time. The investigation presented in the following sections confirms that the classification of projects significantly decreases the number of comparisons.

Table 3 shows the number of comparisons performed in ABE in two different situations: when the classification is not applied and when the projects are classified based on the development type. According to the leave-one-out process, there are $n \times (n - 1)$ comparisons for a dataset consisting of n projects. It is observed that the number of comparisons is reduced by 49 % when the selective classification is considered. This implies that the classification of projects based on the development type can considerably reduce the number of comparisons in ABE and expedite the process of estimation.

Similar to Tables 3 and 4 shows the number of comparisons done in ABE in two different situations: without classification and with classification of projects based on the organization type. In the first situation, all the projects are considered as the historical dataset to be used by the proposed framework while in the second, the projects are classified based on the organization type.

Table 3 The number of comparisons performed in ABE based on the classification of development type

Without classification	Classification	
	New-development	Enhancement
$441 \times 440 = 194,040$	$182 \times 181 = 32,942$	$259 \times 258 = 66,822$
	$32,942 + 66,822 = 99,764$	

As seen in the table, the sum of comparisons performed in five classified sets is significantly less than that for whole dataset (without classification). In this case, the percentage of reduction in number of comparisons is around 75. Therefore, it is confirmed that classification of projects based on the organization type can decrease the number of comparisons done in ABE and speed up the process of estimation.

The effect of classification, based on development platform, on the number of comparisons has been investigated in Table 5. It is observed that the total number of comparisons for whole dataset is considerably larger than that for classified sets. The percentage of reduction is 70, which confirms that the classification of projects based on the development platform can significantly reduce the total number of comparisons performed in ABE.

In general, when the number of projects that exist in the historical dataset is n , the number of comparisons based on the proposed framework will be $n \times (n - 1)$, which is computed based on the leave-one-out technique. If it is assumed

Table 4 The number of comparisons performed in ABE based on the classification of organization type

Without classification	Classification				
	Insurance	Banking	Government	Financial	Manufacturing
$297 \times 296 = 87,912$	$115 \times 114 = 13,110$	$61 \times 60 = 3,660$	$53 \times 52 = 2,756$	$24 \times 23 = 552$	$44 \times 43 = 1,892$
	$13,110 + 3,660 + 2,756 + 552 + 1,892 = 21,970$				

Table 5 The number of comparisons performed in ABE based on the classification of development platform

Without classification	Classification			
	PC	Mid range	Main frame	Multi
$448 \times 447 = 200,256$	$62 \times 61 = 3,782$	$62 \times 61 = 3,782$	$173 \times 172 = 29,756$	$152 \times 151 = 22,952$
	$3,782 + 3,782 + 29,756 + 22,952 = 60,272$			

Table 6 The number of comparisons performed in ABE (in overall)

Without classification	Classification ($k > 1$)			
	Set 1	Set 2	...	Set k
$n \times (n - 1)$	$\frac{n}{k} \left(\frac{n}{k} - 1 \right)$	$\frac{n}{k} \left(\frac{n}{k} - 1 \right)$...	$\frac{n}{k} \left(\frac{n}{k} - 1 \right)$
	$k \times \left(\frac{n}{k} \left(\frac{n}{k} - 1 \right) \right) = n(n - k)/k$			
	If $k > 1$ then $n \times (n - 1) > \frac{n \times (n - k)}{k}$			

that the classification of projects based on the key attributes leads to appearance of k sets so that the size of each set is $\frac{n}{k}$. Therefore, the number of comparisons for each set will be $\frac{n}{k} \left(\frac{n}{k} - 1 \right)$. To sum up, the total number of comparisons for all k sets will be $\frac{n \times (n - k)}{k}$, which is less than $n \times (n - 1)$, for all $k > 1$. Therefore, it can be concluded that the number of comparisons is definitely decreased when the projects are classified based on the key attributes. Table 6 shows the process in which the number of comparisons is investigated.

To further analysis of results, three parameters of MMRE, PRED and comparisons ratio are evaluated for the three

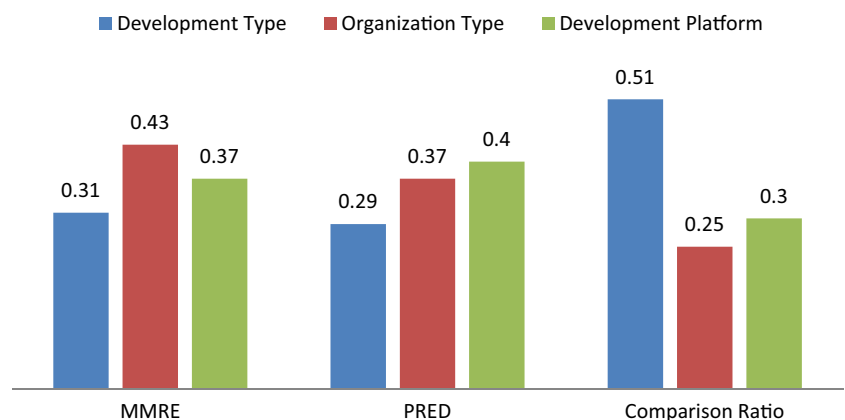
attributes of development type, organization type and development platform, as seen in Fig. 11. The comparison ratio is computed through dividing the number of comparisons after classification by that of all data set. This analysis can be useful to draw a conclusion on influential level of each attribute. From Fig. 11, it is observed that classification based on development type leads to the lowest improvement in both MMRE and PRED.

Moreover, the comparison ratio is in its highest level for the development type attribute. Therefore, it can be concluded that development type is not an appropriate selection in existence of organization type and development platform. Regarding organization type and development platform, making a conclusion is impossible since there is no significant difference between the improvement percentages achieved by these types of classification.

8 Threats to validity

Any types of research may suffer from internal or external threats to validity. In particular, experiment-based studies

Fig. 11 Evaluation of performance parameters in three types of classification



are threatened by a number of issues that must be seriously considered to assure the correct conclusions. In this section, internal and external threats to validity of the current research are elaborated.

A common threat to the validity of experiment-based studies is inaccurate or wrong evaluation process. A very optimistic idea is to consider a part or an entire set of training data as testing set. In such case, the results do not reflect the real performance of the model. Another doubtful assumption is considering a fixed number of unseen data to be evaluated as the testing set. For instance, in the field of effort estimation, a number of projects are selected as the testing set and the remainder as the training set and subsequently a single round of evaluation is conducted to measure model accuracy.

Although this type of evaluation has been frequently used in prior studies, it can simply lead to biased and unreliable results. In contrast, cross validation technique is a widely accepted method strongly recommended by researchers to reveal the real performance of the model. This study used leave-one-out cross validation techniques to avoid any bias in the results and conclusions.

Measurement metrics must be carefully selected according to what the study wants to measure otherwise the direction of the study goes wrong. Since the current study focused on accuracy of estimates, two well-known and widely used metrics, MMRE and PRED(0.25), were utilized to measure the model accuracy. Although there are other measurement metrics, selection of the mentioned metrics makes the results comparable with a wide range of prior studies.

The most important external threat in experiment-based studies is how the proposed model can be generalized in the real world. Any limitations or threats against the application of method must be clarified to avoid incorrect or inappropriate use of the model in future studies. This issue is discussed in the following sections for the method proposed in the current paper. Since ISBSG data repository contains a wide range of software projects collected from more than 24 countries, it is a comprehensive resource to be used as the historical dataset in ABE based models. In this study, a total of 448 projects were selected from ISBSG based upon a series of strict criteria. The number of projects is large enough to support an extensive range of software projects. The selected attributes for the classification process are development type, organization type as well as development platform, which can be identified at early stages of any software project.

The proposed method is able to estimate the effort of software projects whose key attributes are equal to those stated in the proposed framework (in case of ISBSG, two values for development type, five values for organization type and four values for development platform). Since the values of selected attributes used in the comparison of projects are so common among the software projects (e.g. new development, banking, communication, pc, main frame), it can be

said that the proposed method support a large number of software projects in a wide domain. The main aim of this study was to show that the classification of software projects based on important attributes can lead to more reliable results and more extensive support of software projects and consequently the proposed method in its current form along with the selected attributes are able to apply to various types of software projects.

In conclusion, while having a well-defined method for attribute selection considerably increases model adoption, it is not considered as an issue to threaten the use of the proposed method.

9 Conclusion

Analogy-based estimation is a widely accepted method frequently used for software development effort estimation. Despite advantages and capabilities, ABE is unable to overcome the heterogeneous and inconsistent nature of software projects. This shortcoming leads to inaccurate and unreliable estimates. The current paper proposed a framework to evaluate the role of selective project classification in accuracy of estimates achieved by ABE. Development type, organization type and development platform, as three underlying attributes of software projects, were selected to use for the classification purpose. To evaluate the combination of ABE and selective classification, a real dataset that includes 448 software projects was utilized in this study. The classification of projects based on the development type led to two sets of new development and enhancement while the organization type classification led to five sets of insurance, banking, government, financial and manufacturing. PC, Mid Range, Main Frame and Multi were four sets obtained from the development platform classification. According to the proposed framework, the performance of ABE was evaluated on the whole dataset as well as the classified sets using the performance metrics of MMRE and PRED (0.25).

Although it may seem that decreasing the historical dataset size leads to insufficient comparisons in ABE, the results of this paper demonstrated that if classification is performed based on the underlying attributes, the quality of comparisons is increased and the accuracy of estimates is considerably improved. Indeed, the selective classification of projects can increase the consistency and smooth level of historical dataset and decrease the number of biased and irrelevant projects that lead to inaccurate estimates. Besides the increase of accuracy, the selective classification decreases the number of comparisons performed in ABE. Therefore, it can be concluded that considering the nature of software projects (by the selective project classification) in the ABE estimation process increases the estimate accuracy and expedites the estimation process. This can open a window for proposing

insightful models in which the development effort estimation is performed with respect to the underlying attributes of projects. In this paper, real projects were utilized to demonstrate the role of localization in improvement of ABE performance. Although considering real projects helps practitioners to believe the impact of proposed classification, a postmortem analysis is recommended, as well. It can lead to stable conclusions and appropriate roadmap related to ABE use. As future work, we are going to improve the accuracy of estimates achieved by ABE on the classified projects using soft computing techniques.

References

- Albrecht AJ, Gaffney JA (1983) Software function, source lines of codes, and development effort prediction: a software science validation. *IEEE Trans Softw Eng* SE 9(6):639–648
- Angelis L, Stamelos I (2000) A simulation tool for efficient analogy based cost estimation. *Empir Softw Eng* 5:35–68
- Aroba J, Cuadrado-Gallego JJ, Sicilia M-A, Ramos I, García-Barriocanal E (2008) Segmented software cost estimation models based on fuzzy clustering. *J Syst Softw* 81:1944–1950
- Azzeh M (2012) A replicated assessment and comparison of adaptation techniques for analogy-based effort estimation. *Empir Softw Eng* 17:90–127
- Azzeh M, Neagu D, Cowling P (2010) Fuzzy grey relational analysis for software effort estimation. *Empir Softw Eng* 15:60–90
- Bajwa SS (2009) Investigating the nature of relationship between software size and development effort. *Blekinge Institute of Technology*
- Bettenburg N, Nagappan M, Hassan AE (2012) Think locally, act globally: improving defect and effort prediction models. In: 9th Working Conference on Mining Software Repositories. Zurich, Switzerland, pp 60–69
- Chiu NH, Huang SJ (2007) The adjusted analogy-based software effort estimation based on similarity distances. *J Syst Softw* 80:628–640
- Conte SD, Dunsmore HE, Shen VY (1986) *Software engineering metrics and models*. Benjamin-Cummings Publishing Co., Inc, Menlo Park
- Cuadrado-Gallego JJ, Sicilia MA, Garre M, Rodriguez D (2006) An empirical study of process-related attributes in segmented software cost-estimation relationships. *J Syst Softw* 79:353–361
- Cuadrado JJ, Gallego M, Sicilia A (2007) An algorithm for the generation of segmented parametric software estimation models and its empirical evaluation. *Comput Inform* 26:1–15
- Dolado J (2001) On the problem of the software cost function. *Inf Softw Technol* 43:61–72
- Hsu CJ, Huang CY (2011) Comparison of weighted grey relational analysis for software effort estimation. *Softw Qual J* 19:165–200
- Huang SJ, Chiu NH (2006) Optimization of analogy weights by genetic algorithm for software effort estimation. *Inf Softw Technol* 48:1034–1045
- ISBSG (2011) *International Software Benchmarking standard Group*. In
- Jianfeng W, Shixian L, Linyan T (2009) Improve analogy-based software effort estimation using principal components analysis and correlation weighting. In: *International Conference on Software Engineering*. Sun Yat-sen Univ., Guangzhou, China, pp 179–186
- Jingzhou L, Guenther R (2008) Analysis of attribute weighting heuristics for analogy-based software effort estimation method. *Empir Softw Eng* 13:63–96
- Kadoda G, Cartwright M, Chen L, Shepperd M (2000) Experiences using case-based reasoning to predict software project effort. In: *International Conference on Empirical Assessment and Evaluation in Software Engineering*. KEELE University
- Keung J, Kocaguneli E, Menzies T (2013) Finding conclusion stability for selecting the best effort predictor in software effort estimation. *Autom Softw Eng* 20:543–567
- Keung JW, Kitchenham BA, Jeffery DR (2008) Analogy-X: providing statistical inference to analogy-based software cost estimation. *IEEE Trans Softw Eng* 34:471–484
- Bardsiri VK, Jawawi DNA, Hashim SZM, Khatibi E (2012a) Increasing the accuracy of software development effort estimation using projects clustering. *IET Softw* 6:461–473
- Bardsiri VK, Jawawi DNA, Hashim SZM, Khatibi E (2013) A PSO-based model to increase the accuracy of software development effort estimation. *Softw Qual J* 21:501–526. doi:10.1007/s11219-012-9183-x
- Kocaguneli E, Menzies T, Bener A, Keung JW (2012) Exploiting the essential assumptions of analogy-based effort estimation. *IEEE Trans Softw Eng* 38:425–438
- Lafferty MT (2010) *Software effort estimation accuracy: a comparative study of estimation based on software sizing and development methods*. Capella University, Minneapolis
- Li J, Ruhe G (2008) Analysis of attribute weighting heuristics for analogy-based software effort estimation method AQUA+. *Empir Softw Eng* 13:63–96
- Li JZ, Ruhe G (2008) Software effort estimation by analogy using attribute selection based on rough set analysis. *Int J Softw Eng Knowl Eng* 18:1–23
- Li YF, Xie M, Goh TN (2007) A study of genetic algorithm for project selection for analogy based software cost estimation. In: *International Conference on Industrial Engineering and Engineering Management*. Singapore, pp 1256–1260
- Li YF, Xie M, Goh TN (2009a) A study of the non-linear adjustment for analogy based software cost estimation. *Empir Softw Eng* 14:603–643
- Li YF, Xie M, Goh TN (2009b) A study of project selection and feature weighting for analogy based software cost estimation. *J Syst Softw* 82:241–252
- Lin J-C, Tzeng H-Y, Lin Y-T (2012) Automatically estimating software effort and cost using computing intelligence technique. *Comput Sci Inf Technol* 2:381–392
- Mendes E, Watson I, Triggs C, Mosley N, Counsell S (2003) A comparative study of cost estimation models for web hypermedia applications. *Empir Softw Eng* 8:163–196
- Menzies T, Butcher A, Marcus A, Zimmermann T, Cok D (2011) Local vs. global models for effort estimation and defect prediction. In: *26th IEEE/ACM International Conference on Automated Software Engineering*. IEEE Computer Society, Lawrence, KS, pp 343–351
- Milios D, Stamelos I, Chatzibagias C (2011) Global optimization of analogy-based software cost estimation with genetic algorithms artificial intelligence applications and innovations. In: Iliadis L, Maglogiannis I, Papadopoulos H (eds), vol 364. Springer Boston, pp 350–359
- Minku LL, Yao X (2012) Ensembles and locality, insight on improving software effort estimation. *Information and Software Technology*
- Mittas N, Angelis L (2010) Lseba: least squares regression and estimation by analogy. In: *A Semi-Parametric Model for Software Cost Estimation*. *Empirical Software Engineering*, vol 15, pp 523–555
- Posnett D, Filkov V, Devanbu P (2011) Ecological inference in empirical software engineering. In: *26th IEEE/ACM International Conference on Automated Software Engineering*. IEEE Computer Society, pp 362–371

37. Rodriguez D, Cuadrado JJ, Sicilia MA, Ruiz R (2006) Segmentation of Software Engineering Datasets Using the M5 Algorithm. In: 6th international conference on Computational Science—vol Part IV. Springer, Reading, UK, pp 789–796
38. Seo Y-S, Bae D-H (2013) On the value of outlier elimination on software effort estimation research. *Empir Softw Eng* 18:659–698
39. Shepperd M, Schofield C (1997) Estimating software project effort using analogies. *IEEE Trans Softw Eng* 23:736–743
40. Song Q, Shepperd M (2011) Predicting software project effort: a grey relational analysis based method. *Expert Syst Appl* 38:7302–7316
41. Walkerden F, Jeffery R (1997) Software cost estimation: a review of models, process, and practice. In: Marvin VZ (ed) *Advances in Computers*, vol 44. Elsevier, pp 59–125
42. Walkerden F, Jeffery R (1999) An empirical study of analogy-based software effort estimation. *Empir Softw Eng* 4:135–158
43. Wu D, Li J, Liang Y (2013) Linear combination of multiple case-based reasoning with optimized weight for software effort estimation. *J Supercomput* 64:898–918