Using Standard Deviation with Analogy-Based Estimation for Improved Software Effort Prediction

Mohammad Ayub Latif[1]*, Muhammad Khalid Khan[1] and Umema Hani[1]
[1] College of Computing and Information Sciences,
Karachi Institute of Economics and Technology
Karachi, Pakistan
[e-mail: malatif@kiet.edu.pk, khalid.khan@kiet.edu.pk, dr.umema@kiet.edu.pk]
Corresponding author: Mohammad Ayub Latif

Received April 6, 2022; revised November 20, 2022; accepted April 28, 2023; published May 31, 2023

Abstract

Software effort estimation is one of the most difficult tasks in software development whereas predictability is also of equal importance for strategic management. Accurate prediction of the actual cost that will be incurred in software development can be very beneficial for the strategic management. This study discusses the latest trends in software estimation focusing on analogy-based techniques to show how they have improved the accuracy for software effort estimation. It applies the standard deviation technique to the expected value of analogy-based estimates to improve accuracy. In more than 60 percent cases the applied technique of this study helped in improving the accuracy of software estimation by reducing the Magnitude of Relative Error (MRE). The technique is simple and it calculates the expected value of cost or time and then uses different confidence levels which help in making more accurate commitments to the customers.

Keywords: Effort Estimation, Analogy-based estimation, improving accuracy, standard deviation in effort estimation, commitments to customers.

1. Introduction

When planning about the effort or cost estimation of a software four important considerations before selecting an estimation model are the project size, software development style, the development stage and the required accuracy. The size can be classified as small, medium and large, different estimation models have their defined values for the size; the development style can be sequential, iterative or coupled. The development stage refers to the time in the lifecycle when an estimator is estimating the project; it can be the start of the project which is early requirements, to middle or late. The fourth consideration is the accuracy which an estimator is targeting;

After getting the outputs from an estimation model, data is needed for calibrating it into meaningful estimates. All models require data and, in his book, Steve McConnell has identified three types of data. Industrial data is the data of other organizations, historical data is the data of the same organization of previous projects and project data is the data of the project which is estimated. One requirement for the use of project data is that, the project which needs to be estimated should follow the iterative development life cycle, so the data of the first iteration can be used for the calculations of later iterations [1].

Researchers have also shown that least accurate results are from calibration done with industrial data, the historical data gives better results than the industrial data and the most accurate results with lowest variance is by the use of the project data [2]. Project managers need to check that completion time for a task is given intelligently so that Parkinson's Law does not apply in their on-going projects. Parkinson's Law states that work generally takes up all the time which is allocated for a task. So, if you give your developers, four days to finish a one-day task, it is expected that the task will now acquire four days [3][4]. Mostly in the modern era we have dynamic estimation models than compared to the flat models in which the number of team members can vary with respect to the different phases of the SDLC. With a dynamic estimation model the team size can be of 2 people in the requirement phase and 10 people in the development phase [1][5][2].

Broadly software estimation models are divided into two different categories algorithmic and non-algorithm model. The popular models in algorithm models are Lines of Code (LOC), Function points (FP) and Constructive Cost model (COCOMO). The non-algorithmic models comprise of expert judgment, analogy-based techniques, proxy techniques and pricing to win.

Estimation for defects through a defect prediction mechanisms for software with identification of challenges for defect prediction [6] shows the use of estimation which is other than cost, time and effort. The control of software activities and predicting about when a development will end is a difficult task, in order to adapt changes, some researchers have proposed a generalized software reliability model that is based on stochastic process to stimulate the software development that includes uncertainty [7]. Another study has explored the possibilities of application of Artificial Neural Network (ANN) as a tool for predicting software development effort. It proposed an ANN model for predicting software development effort [8]. In another work a systematic review of software effort estimation models built using ML techniques. All the empirical studied published in the time period of January 1991 to December 2017 were considered in the review. The work concludes that support vector machines (SVM) and regression techniques in combination are characterized by better predictions when compared with other Machine learning and non-Machine learning techniques [9]. It is important to note that metrics not only pertain to software costing and estimation; product metrics usage can lead towards better software quality. A study has proposed new technique for the visualization of metrics which will ultimately help in improving the software

quality [10].

Generally, most of the project managers know that there is no best effort estimation model or method that can be applied to a particular case of estimation, if the client is forcing for a low-cost solution this can also lead to an overrun. Other important known concept is that estimations are often misleading [11]. It is well understood that Software Process Improvement can occur if we move towards better estimation for software. In recent times the concept of Global Software Engineering (GSE) has also emerged and many organizations are involved in Global Software Development (GSD). A systematic literature review is performed on success factors and barriers to software process improvement for Global Software Development (GSD) [12]. The concept of GSD has also given rise to offshore software development where low-cost countries are used for developing software for another country. A study has identified the challenges for managing offshore contracts from the vendors' perspectives [13].

The core idea of analogy-based effort estimation (ABEE) is that you can create the estimate of a new project by comparing it with the estimates of an old project which has already been accomplished by your organization. ABEE or estimation by analogy comprises of 4 major steps for calculating the effort of the software as shown in **Fig. 1**:

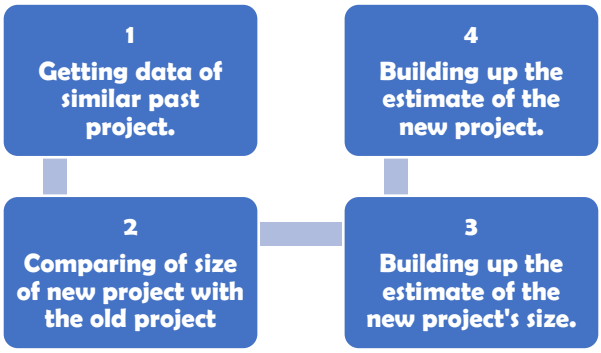

Fig. 1. Steps for Estimation by analogy.

The pioneers of introducing analogy based estimates were Shepperd and Schofield and they proposed this as a non-algorithm model for software effort estimation [14]. There are a few constraints which are mandatory for the accuracy of estimation by analogy; the first consideration is that the size of the previous and the current project should not vary on a larger scale. The development technologies for the both the projects should be same, this means if a project is to be developed in C# language, we cannot use a project developed in C Language as the baseline project. The difference between the team sizes of the new and the old project should also be minimal. Another important constraint is that the type of the projects should be same, a system software cannot be compared to form an estimate of an information system [1].

Following are the major contributions of the present study:

1. Detailed review of analogy-based effort estimation techniques in recent years and how different improvements have been suggested in them to achieve better accuracy.
2. A simplified real case study that shows the effort estimation of a software in a simplified way.

3. Very simple technique of standard deviation applied to the initial calculated effort in order to achieve better accuracy.
4. Validation of achieving improvement through the proposed approach over an available industrial dataset from different software houses.

The rest of the paper is structured as follows, in section 2 we provide the related work which focuses on the latest research trends related to analogy-based effort estimation. In section 3 we present a simple case of estimation by analogy and show the calculation of effort in terms of persons-month and we also recommend how the estimation by analogy can be improved by using standard deviation. We also apply our standard deviation methodology on an available dataset for agile software. In section 4 we provide our results and discussion and in section 5 we conclude our paper with directions for future research.

2. Related Work

In this section we look into all the work that has been carried out related to analogy-based estimation, we investigate the variants of analogy-based estimation and how improvements to the traditional methods had been shown by different researchers.

A systematic mapping of ASSE papers from 1990 to 2012 has been performed. The research objectives were to identify the studies with respect to the estimation accuracy, comparison of accuracy, context of the estimation, ASSE tools and impact of techniques which were used in combination to ASSE method [15].

To find improvements in ASSE technique a domain of review comprised of 24 papers which were selected through a formal tough process. The results show that improvement of ABE can be performed through adjustment, grey theory, attribute weighting and attribute selection techniques [16].

Analogy based estimation (ABE) is criticized because of low prediction accuracy, the large memory requirement and the expensive computation cost. To provide a solution for these problems a project selection technique for ABE (PSABE) is proposed which reduces the whole project base into a small subset that consist only of representative projects. Finally, PSABE is combined with the feature weighting to form FWPSABE for a further improvement of ABE. To validate the methods four datasets are used (two real-world sets and two artificial sets) and compared with conventional ABE, feature weighted ABE (FWABE), and machine learning methods. The results conclude that project selection technique could significantly improve analogy-based models for software cost estimation [17].

A work has investigated non-uniform weighting through kernel density estimation. After an extensive experimentation of 19 datasets, 3 evaluation criteria, 5 kernels, 5 bandwidth values and a total of 2090 ABE variants, it concludes that non-uniform weighting through kernel methods cannot outperform uniform weighting ABE [18].

A novel technique is proposed that relies on reasoning by analogy, fuzzy logic and linguistic quantifiers for estimating effort, provided that the software project is represented either by categorical or numerical data. Use of fuzzy logic-based cost estimation models is more suitable if unclear or inaccurate information are considered [19].

In a work to rank the adaptation techniques of analogy-based estimation a comparison of eight different ranking techniques for analogy-based estimation using larger datasets concludes that linear adaptation techniques outperform all other techniques [20].

To achieve accuracy and as to the fact that no estimation model outperforms other models in all situations, the importance of estimating from ensembles of various single technique. A

work proposes similar ensembles based on single classical analogy and single fuzzy analogy. Experiments were conducted across seven datasets, that concludes that fuzzy analogy ensembles achieved better performance than classical analogy ensembles [21].

Ibtissam Abnan et. al. used missing data techniques with fuzzy analogy. They found that Pred (0.25) and Standardized Accuracy (SA) measure different aspects of technique performance. They suggest that SA should not be used alone to conclude about a technique's accuracy and other metrics should also be involved with it and they recommend the involvement of Pred (0.25) as the other metric [22].

A Squares Support Vector Machine (LS-SVM) method that is nonlinear adjustment method is used for calibration. The work tested it on some datasets and compared it results with artificial neural network (ANN) and extreme learning machines (ELM) [23].

To overcome the errors related to analogy-based estimation, a work shows that S-membership function can be used to overcome the problems of an estimator to select the right set of projects to reach to a comparison [24].

Analogy-based estimation is built upon the principle of case-based reasoning (CBR) based on the k similar projects completed in the past. Therefore, the determination of the k value is crucial to the prediction performance. The researchers have worked and proposed a technique that uses hierarchical clustering in order to produce a range for k through various cluster quality criteria [25].

A research has compared six similarity measures for analogy-based estimation, it concludes that Euclidean and Manhattan similarity measures gives more accurate result in estimation for the datasets of software projects [26].

A work finds out, that instead of keeping all the historical data for COCOMO, using recently completed projects data of shorter duration will help in more accurate results in estimation. Similarly, k-nearest neighbors will also produce accurate results for Estimation by analogy [27].

Achieving accuracy in projects where the size of the current project is different to the completed past projects relies on effort adaptation. The work performs systematic comparison of effort estimators that were optimized by Bayesian optimization techniques. The experiment was carried out on 13 standard datasets. It concludes that a model which integrates gradient boosting machine algorithm has out-performed all other techniques [28].

A new analogy-based approach is proposed named as 2FA-kprototypes that can be utilized when both kind of attributes are involved. It used some datasets to compare the accuracy of 2FA-kprototypes with the traditional analogy-based estimation and 2FA-kmodes (this technique was developed in their earlier research). The verification results showed that 2FA-kprototypes and 2FA-kmodes both techniques performed better than traditional analogy-based effort estimation [29].

Where software projects are defined by a combination of continuous and categorical features; in a work an improvement is made to the 2FA-kprototypes techniques by using the 2FA-cmeans. This new techniques uses a fuzzy c-mean clustering technique that cluster objects which have mixed attributes. This 2FA-cmeans was tested on 6 different datasets and it outperforms their previous 2FA-kprototype technique and also all other classical analogy techniques [30].

A new solution function has been proposed to improve the estimation accuracy of Analogy based estimates. The function is called SABE (Stacking Regularization in analogy-based software effort estimation. The crucial point about SABE is stacking which is a machine learning technique. Stacking works on multiple models and combines the capabilities of all in order to better predict the estimate. Four different datasets are used for validation and results

suggested that SABE's performance is better than the former studies [31].

A study has investigated the effect of the LEM algorithm on optimization of features weighting and have proposed a new method. They checked the effectiveness of the algorithm on two datasets, Desharnais and Maxwell. They used evaluation metrics like MMRE, PRED (0.25), and MdMRE to evaluate and compare the proposed method against previous algorithms. Their technique show considerable improvement in estimating the cost of the software [32].

As analogy-based estimation requires prediction of the best number of analogies and adjustment technique selection for achieving the best possible estimates, a work has proposed a new adjusted ABE model for optimization and approximation of complex relationships between different features. It shows that the use of this model has improved the performance of ABE [33].

A proposed estimation model known as the Fuzzy Analogy based Software Effort Estimation model (FASEE) makes successful use of fuzzy logic with approximate reasoning theory to handle imprecision and uncertainty. In a recent work enhancement has been made to the FASEE model and problems related to the low quality of data and uncertainty in the reasoning process are solved to some extent. This new model is compared in thirteen software project datasets and it is concluded that the model performs better in terms of accuracy. The model is named as Consistent Fuzzy Analogy-based Software Effort Estimation (CFASEE) [34].

The shortcomings of Analogy-Based estimation tools are identified and a new enhanced model for analogy-based estimation is proposed. A system prototype is also prepared which is called EffortEst and it is based on the enhanced model. The authors have shown that EffortEst provides the nearest best estimation and the user intervention is also minimal [35].

A new framework is proposed that uses case-based reasoning (CBR) model along with considering the comprehensive set of requirements that includes the functional, non-functional requirements both along with the domain properties. The framework is tested on a set of thirty-six students projects and shows that the difference in terms of calculated and actual effort was in the range of 10% [36].

International Software Benchmarking Standards Group (ISBSG) dataset is used in a study to confirm that the usefulness of applying linguistic values rather than the numerical values in analogy-based estimation can bring much better results in terms of accuracy[37].

The **Table 1** below shows the references of the work carried out for bringing an improvement in the analogy-based estimates. The **Table 1** headers are restricted to paper reference, pros and cons, the accuracy metric used and the details of the accuracy in the last column. Only those studies are entered in the **Table 1** from the reported studies which tested the analogy-estimation improvement technique and validated it by using some accuracy metrics.

Table 1. Comparative analysis of existing approaches for improvement in ABEE

Paper ID	Proposed Technique Name	Pros	Cons	Accuracy Metric	Accuracy
[17]	Feature Weighing Project Selection Technique for Analogy-	Promising result that it can significantly improve the analogy-based estimation even the ones which	Comparatively smaller memory requirement because of only selected projects. Only tested for	Mean Magnitude Relative Error (MMRE) Pred (0.25)	For both the real and the artificial datasets the proposed technique based on

	based estimation.	already applied feature weighing. Lesser computation cost than compared to traditional analogy-based estimation.	projects developed using the waterfall methodology.	Median Magnitude of Relative Error (MdMRE)	feature weights project selection outperformed all other techniques in most cases which was validated by using accuracy metrics. For the real dataset their proposed techniques achieved the best testing performance (0.32 for MMRE, 0.44 for PRED (0.25) and 0.29 for MdMRE)
[37]	Fuzzy Analogy with Linguistic values.	Simple explanation of classical analogy and fuzzy analogy is provided in the work. Promising results for Fuzzy analogy than compared to classical analogy.	Diverse datasets from different applications domains are not accommodated. Datasets with high quality ratings were included in the experiment and datasets classified of lesser quality were ignored. The technique was only tested for new development, this leaves a question mark for using the technique for the enhancements of existing projects or redevelopment.	Pred (20)	For All-in method for evaluation the values of Pred (20) are 50 and 76.35 using Classical and Fuzzy Analogy respectively; For Jackknife method, the values of Pred (20) are 16.89 and 24.32 using Classical and Fuzzy Analogy respectively. In both the cases there is improvement through the Fuzzy Analogy technique.

[19]	Cost Estimation based on soft computing techniques.	Block diagram is created for the techniques which makes it easy to understand the proposed technique.	MMRE calculation was made only for one dataset, although the work utilized three datasets in the experiment. The datasets are old and surely does not accommodate the software which are developed using the newer methodologies of software development.	Mean Absolute Relative Error (MARE) Mean Magnitude of Relative Error (MdMRE)	The MMRE for the COCOMO NASA Dataset reduced to 2.6 percent with the proposed method than compared to two other methods where the calculated MMRE was 32.65 percent and 56.46 percent respectively.
[21]	Fuzzy Analogy Ensembles.	The work provides an investigation of fuzzy and classical analogy ensemble technique used for effort estimation. The study answers four research questions that relates to estimation through ensembles for analogy.	The work only incorporated numerical attributes and categorical data was missed out.	Standardized Accuracy (SA) based on Mean Absolute Error (MAE)	Fuzzy Analogy ensembles often outperform Classical Analogy ensembles in terms of SA even though whatever combiner rule and number of solo techniques are used to construct the ensembles.
[25]	Hierarchical clustering for Analogy-based effort estimation.	The research work pointed towards the problem domain for identifying the clustering technique on the basis of datasets, i.e. which clustering technique will be feasible for which kind of a dataset.	For Random k only one other technique (Baker's Technique) was evaluated with the proposed technique although many other techniques exist for keeping a random value of k.	Mean Magnitude Relative Error (MMRE)	They tested their proposed technique of hierarchical clustering on six datasets and compared it with other variants of dynamic k size and fixed k size. Out of the 6 datasets the MMRE was improved in three cases using the

					proposed approach. For the Desharnais dataset they reported the lowest MMRE at 0.492 than compared to the other four techniques.
[28]	Effort adaptation using gradient boosting machine algorithm	Highly accurate and intuitive proposed model that does not require any expert interactions. 10 effort estimation techniques are compared in the study.	Not tested for search-based effort estimation techniques.	The study used the losses metric based on six different accuracy metrics used for effort estimation.	The loses were counted at minimum for the proposed technique and all the 13 datasets which incorporated the proposed technique had losses at the lowest group i.e. is from 0% to 5%. None other compared technique could categorize all the datasets in the lowest losses group.
[32]	Learning Objective Model with Analogy-Based Estimation (LEMABE) .	The study has combined the Learning Evolution model with analogy-based estimation, a flowchart algorithm is also provided for the technique.	N/A	Mean Magnitude Relative Error (MMRE) Pred (0.25) Median Magnitude of Relative Error (MdMRE)	Of the 3 different datasets used for validating the technique for the Deshernais dataset, it produced the best result for all the three different accuracy metrics than compared to all other techniques. The results were MMRE = 0.21, Pred (0.25) = 0.60,

					and MdMRE = 0.15. In al other datasets also, there was significant improvement by utilizing the proposed technique.
[35]	EffortEst (Effort Estimation)	Critical analysis of estimation models is performed to identify the short comings. User intervention is limited with the proposed technique and they have also developed a prototype for the estimation tool.	The EffortEst is compared with only three other estimation tools, although there are numerous estimation tools available for effort estimation. Lack of evidence for claiming that ESTOR with which they compared their estimation tool is the best.	Persons-month	The calculated effort through EffortEst in the study was closest to the estimate calculated through a tool which they claimed as the best estimation tool and the name of the tool with which they compared EffortEst is ESTOR.
[36]	Multi-criteria decision technique.	Uses incremental approach for software prediction at an early stage of software development.	The work only relied on students' projects for validation, this means no real world or industrial dataset was used for calculating the accuracy.	Pred (0.10)	The work has validated the proposed methodology on 36 students' projects and they used the predictability metric at 10%. 97% of the total projects met the pred (0.10) criteria.

The section 3 presents a case study based on analogy-based estimate and then also shows how it can be improved using the standard deviation technique which we have proposed. The best thing about our approach is we have provided a complete case study step by step. Unlike in most of the studies the core advantage we have in our dataset is that it is based on the agile methodology. Secondly our dataset is collected from six different software houses of Pakistan.

3. Analogy-Based Estimation Technique

In this section we present a small and a simple case so that the readers can have an idea as to how the analogy-based estimation works. Let's suppose an organization has recently created a project in which they have worked on a biometric system for marking attendance of employees of a company and created a web application for their HR team including charts for performance analysis of employee's office timing and an android application for upper management to watch the data and keep an eye on employees attendance.

Now a new client needs a similar Attendance system with a little difference. In the new case some employees of the company work on client sites, while others work in office. So, the attendance could not be marked using the thumb machine as it is installed at company's office and not at the client's location. Considering this problem, the new solution is an android application that will provide login for the employees of that company and when those employees will enter the client's premises, the location service will be used to mark their attendance.

The organization has chosen estimation by analogy for this project as this project is almost similar to previous project. For the initial step the **Table 2** shows the previous project's modules, their LOCs, number of features of previous and new project. Finally, it also shows the multiplication factor used for the new project calculated from the details of previous and new project.

Table 2. LOC of previous similar project and the current project

Modules (LOCS)	Previous Project Detail LOC	Previous Project Detail	Current Project	Multiplication Factor	Current Project LOC
Requirement Analysis	8000	60 features	90 features	1.5	12000
MySQL Tables	17500	20 tables	25 tables	1.25	21875
Android Part	6000	6 activities	12 activities	2	12000
Graphs and Charts	2000	5 charts	10 charts	2	4000
Java Code	30000	80 classes	90 classes	1.125	33750
APIs	4000	6 APIs	9 APIs	1.5	6000
Hardware Integration	20000	40 classes	48 classes	1.2	24000
SQL Classes	6000	12 classes	12 classes	1	6000
Total	**93500**				**119625**

In **Table 2**, the multiplication factor is calculated by dividing the previous project features with the new project features. The LOC of the new project is calculated by simply multiplying the previous project's LOC with the multiplication factor.

Using the LOCs of the previous and current project the size ratio is calculated and it is shown that the effort for the current project is estimated at 64 staff-months than compared to the effort of 50 staff-months of the previous project. **Table 3** shows the details.

Table 3. Effort of the new project in staff-months

Term	Value
Current Project LOC	119625
Previous Project LOC	93500
Size Ratio	1.28
Previous Project Effort	50 staff-months
Current Project Effort	64 staff-months

3.1. Improving the Analogy-Based Estimation

This subsection shows the use of simple standard deviation technique to get better confidence for the estimate that was calculated in the last section. Standard deviation technique requires worst case, best case and most likely estimates. To incorporate this from the previous multiplication factor for analogy estimates, we will have three values, one for the best case, the other for the most likely and the last for the worst case. From the three values we will generate the expected value as suggested by Putnam[38]. The **Table 4** below shows the best case, worst case and the most likely values of the previously calculated multiplication factor. Our new estimation case incorporating the best and the worst cases is given in the **Table 4**.

Table 4. Best- and Worst-case LOC calculation of the new project

MF: Multiplication Factor

Modules (LOCS)	Previous Project LOC	MF (Best Case)	MF (Most Likely)	MF (Worst Case)	Current Project LOC (Best Case)	Current Project LOC (Most Likely)	Current Project LOC (Worst Case)
Requirement Analysis	8000	1	1.5	2	8000	12000	16000
MySQL Tables	17500	0.80	1.25	1.4	14000	21875	24500
Android Part	6000	1.25	2	2.40	7500	12000	14400
Graphs and Charts	2000	1.5	2	2.75	3000	4000	5500
Java Code	30000	1	1.125	1.75	30000	33750	52500
APIs	4000	1	1.5	1.80	4000	6000	7200
Hardware Integration	20000	0.8	1.2	1.80	16000	24000	36000
SQL Classes	6000	0.5	1	1.25	3000	6000	7500
Total	**93500**				**85500**	**119625**	**147600**

In **Table 4** we also calculate the best case, worst case and most like size of the software in terms of LOC.

Using the approach of calculating the effort in terms of staff-months, now when we use the values of LOC best case, LOC most likely and LOC worst case. Effort for all the three cases is shown in **Table 5**.

Table 5. Best and worst and most likely effort in persons-month

Best Case Effort	Most Likely Effort	Worst Case Effort
46 staff-months	64 staff-months	79 staff-months

A known way in statistics is to assume that the one sixth of a difference between a maxima and minima is equal to one standard deviation. This way of calculating the standard deviation assumes that the maxima include 99.86% percent chances of meeting the estimate and the minima holds 0.135% percent chances[1]. As defined the standard deviation will be the difference between maxima and minima divided by 6. In our case it will be 79-46/6=33/6=5.5. And our expected case will be based on the Expected value formula which is shown in Equation (1).

$$Expected\ Value = \frac{[BestCase+(4\ x\ MostLikelyCase)+WorstCase]}{6} \quad (1)$$

In our case the Expected Value is 63.5 which is almost 64 staff-months. So, in our case our most likely value is also our expected value. In order to be more confident on our calculated effort value, we see the percentage confident statistically valid values for 70%, 80% and 90% percentage confident calculations in the **Table 6**. The values are also given in the book by Steve McConnell[1]. The standard deviation was 5.5 as calculated previously.

Table 6. Estimated Effort with different percentage confidence

Percentage Confident	Calculation	Effort Estimation
70%	Expected Case + (0.52 x SD)	66.86=67 staff-months
80%	Expected Case + (0.84 x SD)	68.62=69 staff-months
90%	Expected Case + (1.28 x SD)	71.04=71 staff-months

As shown in the **Table 6**, committing 71 staff-months as effort for our system will have 90% confidence that the project will not take more than 71 staff-months, similarly 67 staff-months and 69 staff-months will have 70% and 80% confidence respectively. By incorporating this standard deviation in analogy-based estimation, there are more chances of meeting our commitment that will lead to better level of customer satisfaction.

To strengthen our approach, we have used this mechanism in a dataset for agile based estimation. As agile based development uses an iterative and incremental approach, expert judgment is also used for effort estimation in agile based software system.

Agile based estimation generally uses the analogy-based estimation therefore we believe it's a good choice to implement this mechanism on an agile based dataset. The authors in their research work have identified through the systematic literature review and survey that one of the most frequently estimation technique for agile based development is estimation by analogy, they have nicely classified the estimation techniques for agile based software development [39].

The dataset used for using standard deviation approach in order to improve the effort estimation contains data of 21 previous projects developed using Scrum-based agile software development. The dataset has been collected from [40] which claims that the data was first collected from six different software houses of Pakistan. The dataset contains 21 instances and 9 attributes. Each instance represents one project related data which provides information including Sprint Size, monthly Work Days, Team's Initial Velocity (Vi), total Time for completing one sprint, total Cost spend on one sprint, Efforts completed by the team in one sprint, monthly Team Salary, Dynamic Force Factor (D) and Teams' Final Velocity (V). In order to show the use of standard deviation on the dataset we will use the values of actual time and estimated times as shown in **Table 7**. The **Table 7** only shows that values that we will be using for our case.

Table 6. Dataset of 21 software developed using agile methodology

No.	Actual Time	Est. time
1	63	58
2	92	81
3	56	52
4	86	87
5	32	29
6	91	95
7	35	29
8	93	84
9	36	35
10	62	66
11	45	41
12	37	39
13	32	35
14	30	26
15	21	22
16	112	103
17	39	40
18	52	50
19	80	76
20	56	51
21	35	34

In order to incorporate the standard deviation, we have first calculated the best case and worst case from the estimated time. As we did not have the actual best case and worst-case figures, we just subtracted 20% from the estimated time for the best case and added 20 percent to the estimated time for the worst case. We assumed the estimated time as the most likely value, using these three values we calculated the expected value as discussed in the example previously. As same percentage was used for generating the best and the worst case, the expected value is same as the estimated value given in the dataset. But this approach helped us in calculating the standard deviation with the help of which we were able to generate more than one estimated time with different level of confidence. In **Table 8** we show our standard deviation part on the dataset and also show the new estimated time with 70, 80 and 90% confident levels.

Table 8. Standard Deviation and different confidence levels of time

No.	Est. time	Est. time (ML)	WC	BC	EV	SD	70%	80%	90%	Actual
1	58	58	46.4	69.6	58	3.86	60.01	61.24	62.94	63
2	81	81	64.8	97.2	81	5.4	83.80	85.53	87.91	92
3	52	52	41.6	62.4	52	3.46	53.80	54.91	56.43	56
4	87	87	69.6	104.4	87	5.8	90.01	91.87	94.42	86
5	29	29	23.2	34.8	29	1.93	30.0	30.62	31.47	32
6	95	95	76	114	95	6.33	98.29	100.3	103.1	91
7	29	29	23.2	34.8	29	1.93	30.0	30.62	31.47	35
8	84	84	67.2	100.8	84	5.6	86.91	88.70	91.16	93
9	35	35	28	42	35	2.33	36.21	36.96	37.98	36
10	66	66	52.8	79.2	66	4.4	68.28	69.69	71.63	62
11	41	41	32.8	49.2	41	2.73	42.42	43.29	44.49	45
12	39	39	31.2	46.8	39	2.6	40.35	41.18	42.32	37
13	35	35	28	42	35	2.33	36.21	36.96	37.98	32
14	26	26	20.8	31.2	26	1.73	26.90	27.45	28.21	30
15	22	22	17.6	26.4	22	1.46	22.76	23.23	23.87	21
16	103	103	82.4	123.6	103	6.86	106.5	108.7	111.7	112
17	40	40	32	48	40	2.66	41.38	42.24	43.41	39
18	50	50	40	60	50	3.33	51.73	52.8	54.26	52
19	76	76	60.8	91.2	76	5.06	78.63	80.25	82.48	80
20	51	51	40.8	61.2	51	3.4	52.76	53.85	55.35	56
21	34	34	27.2	40.8	34	2.26	35.17	35.90	36.90	35

The first column of the **Table 8** is the project number, the second column is of the estimated time as given in the dataset, the third column is the estimated time which we have assumed as most likely for our expected value. The fourth and the fifth columns are for the best and the worst case which are calculated by adding and subtracting 20% from the estimated values

respectively. The sixth column is the expected time generated through the expected value formula given in equation 1. The seventh is the standard deviation calculated by dividing the difference of worst and the best case with 6. Eighth, ninth and tenth columns are the estimated time of the project incorporating the standard deviation on 70, 80 and 90 percent confidence levels respectively. The eleventh which is the last column is the actual time that was spent on the project.

In the next section we calculate the Magnitude of Relative Error for all of our estimated cases and provide our understanding of the results. The Mean Magnitude of Relative Error (MMRE) is also calculated.

4. Results and Discussion

This section shows the Magnitude of Relative error (MRE) of 4 cases, first for the previous estimated time value from the dataset, then the MRE for our estimates with 70, 80 and 90% confidence levels which are shown in **Table 9**. The MRE was calculated using the formula given in Equation (2).

$$MRE = AbsoluteValue\ x\ [\frac{ActualResult - EstimatedResult}{ActualResult}] \quad (2)$$

Table 9. MRE of actual dataset using the actual time of project completion, 70%, 80% and 90% confidence time

No.	MRE (Original Dataset)	MRE with 70% confidence	MRE with 80% confidence	MRE with 90% confidence
1	7.93	4.74	2.78	0.08
2	11.95	8.90	7.02	4.44
3	7.14	3.92	1.94	0.78
4	1.16	4.66	6.82	9.79
5	9.37	6.23	4.30	1.64
6	4.39	8.01	10.24	13.30
7	17.14	14.27	12.50	10.07
8	9.67	6.54	4.61	1.96
9	2.77	0.59	2.66	5.51
10	6.45	10.14	12.41	15.53
11	8.88	5.73	3.78	1.11
12	5.40	9.05	11.30	14.40
13	9.37	13.16	15.50	18.70
14	13.33	10.32	8.48	5.93
15	4.76	8.39	10.62	13.70
16	8.03	4.84	2.88	0.188
17	2.56	6.11	8.30	11.31
18	3.84	0.51	1.53	4.35
19	5	1.70	0.32	3.10

20	8.92	5.77	3.82	1.15
21	2.85	0.51	2.58	5.43

Table 9 shows that from the total 21 cases, the MRE has improved in 14 cases when standard deviation was applied. The MMRE (Mean Magnitude of Relative Error) has also improved in all the three cases with 70%, 80% and 90% confidence levels. This is shown in **Table 10**.

Table 10. MMRE of actual dataset using the actual time of project completion and MMRE with 70%, 80% and 90% confidence levels

MMRE (Original Dataset)	MMRE with 70% confidence	MMRE with 80% confidence	MMRE with 90% confidence
7.19	6.38	6.40	6.78

The **Fig. 2** shows the MRE of all the kinds of data, the first graph is of the actual MRE from the dataset, and the next three graphs show the MRE of the actual dataset with the MRE of our 90, 80 and 70% confidence cases respectively.

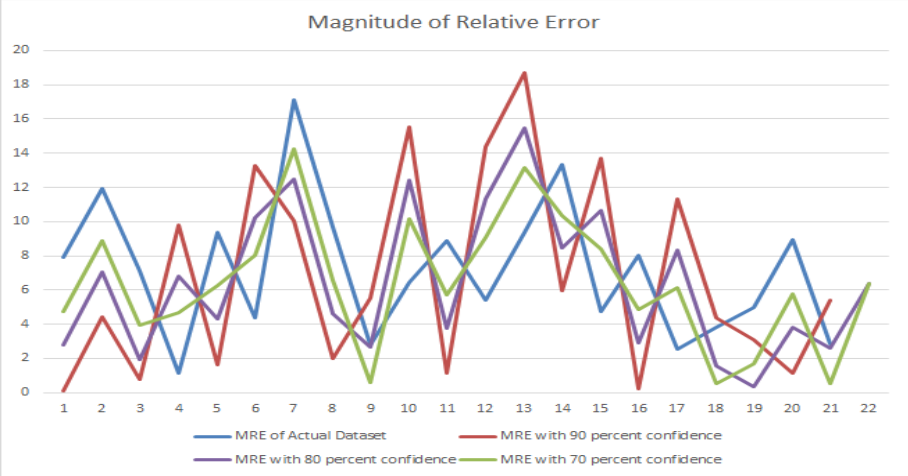

Fig. 2. MRE graphs of the original dataset and three standard deviation based confident levels

It is interesting to observe that when the original MRE is less, means it is 5% or less then the MRE for the standard deviation has gone up and not improved the results. We believe that this dataset is very good in terms of estimation results as the difference between estimated value and actual value in terms of time is very less. This can be known from the fact that the highest MRE in the 21 projects' dataset is 11 percent. Although in estimation it is believed that 25% difference in actual and estimated in also considered good[1], so we can say that this dataset is exceptionally good. Considering the improvement is 14 cases in overall 21 cases we believe that our standard deviation-based technique will help in achieving more accurate estimates in other datasets where the MRE is ten percent or more, so we expect better results in all estimation cases.

5. Conclusion and Future Direction

In this paper, we showed a case of analogy-based estimation on a software system; in order to improve the estimate, we applied simple standard deviation on the estimate. We conclude that calculating estimates with standard deviation will give more confidence while committing completion time to the customers. To strengthen our case, we used the same standard deviation methodology on an available dataset of agile software. Out of 21 instances where the actual time and estimated time was already given in the dataset our methodology helped in improving 14 cases. This means an improvement in 66% of the cases which we showed through the calculation of magnitude of relative error. We conclude that standard deviation should always be applied to the estimates that are generated in order to gain more confidence and better chances of accuracy.

In future we plan to use standard deviation methodology on other datasets also; this can also be applied to cases where software effort is calculated by using other techniques rather than analogy-based estimation.

References

[1] S. McConnell, *Software estimation: demystifying the black art*, Microsoft press, 2006.
[2] M. A. Latif, M. Y. Khan, and K. Bashir, "Practices for Achieving Accuracy in Software Costing and Estimation," *KIET Journal of Computing and Information Sciences*, vol. 1, no. 1, pp. 83–95, 2018.
[3] C. N. Parkinson and R. C. Osborn, *Parkinson's law, and other studies in administration*, vol. 24. Houghton Mifflin Boston, 1957.
[4] C. F. Kemerer, "An empirical validation of software cost estimation models," *Communications of the ACM*, vol. 30, no. 5, pp. 416–429, 1987. Article (CrossRef Link).
[5] B. Boehm, C. Abts, and S. Chulani, "Software development cost estimation approaches - A survey," *Annals of software engineering*, vol. 10, no. 1–4, pp. 177–205, 2000. Article (CrossRef Link).
[6] Z. Li, X.-Y. Jing, and X. Zhu, "Progress on approaches to software defect prediction," *IET Software*, vol. 12, no. 3, pp. 161–175, 2018. Article (CrossRef Link).
[7] K. Honda, H. Washizaki, and Y. Fukazawa, "Generalized software reliability model considering uncertainty and dynamics: Model and applications," *International Journal of Software Engineering and Knowledge Engineering*, vol. 27, no. 06, pp. 967–993, 2017. Article (CrossRef Link).
[8] Y. Singh, A. Kaur, P. K. Bhatia, and O. Sangwan, "Predicting software development effort using artificial neural network," *International Journal of Software Engineering and Knowledge Engineering*, vol. 20, no. 03, pp. 367–375, 2010. Article (CrossRef Link).
[9] A. Ali and C. Gravino, "A systematic literature review of software effort prediction using machine learning methods," *Journal of Software: Evolution and Process*, vol. 31, no. 10, p. e2211, 2019. Article (CrossRef Link).
[10] R. Ishizue et al., "Metrics Visualization Techniques Based on Historical Origins and Functional Layers for Developments by Multiple Organizations," *International Journal of Software Engineering and Knowledge Engineering*, vol. 28, no. 01, pp. 123–147, 2018. Article (CrossRef Link).
[11] M. Jørgensen, "What we do and don't know about software development effort estimation," *IEEE software*, vol. 31, no. 2, pp. 37–40, 2014. Article (CrossRef Link).
[12] A. A. Khan and J. Keung, "Systematic review of success factors and barriers for software process improvement in global software development," *IET software*, vol. 10, no. 5, pp. 125–135, 2016, Article (CrossRef Link).
[13] S. U. Khan and A. W. Khan, "Critical challenges in managing offshore software development outsourcing contract from vendors' perspectives," *IET software*, vol. 11, no. 1, pp. 1–11, 2017. Article (CrossRef Link).

[14] M. Shepperd and C. Schofield, "Estimating software project effort using analogies," *IEEE Transactions on software engineering*, vol. 23, no. 11, pp. 736–743, 1997. Article (CrossRef Link).
[15] A. Idri, F. azzahra Amazal, and A. Abran, "Analogy-based software development effort estimation: A systematic mapping and review," *Information and Software Technology*, vol. 58, pp. 206–230, 2015. Article (CrossRef Link).
[16] V. K. Bardsiri, D. N. Abang Jawawi, and E. Khatibi, "Towards improvement of analogy-based software development effort estimation: A review," *International Journal of Software Engineering and Knowledge Engineering*, vol. 24, no. 07, pp. 1065–1089, 2014. Article (CrossRef Link).
[17] Y.-F. Li, M. Xie, and T. N. Goh, "A study of project selection and feature weighting for analogy based software cost estimation," *Journal of Systems and Software*, vol. 82, no. 2, pp. 241–252, 2009. Article (CrossRef Link).
[18] E. Kocaguneli, T. Menzies, and J. W. Keung, "Kernel methods for software effort estimation," *Empirical Software Engineering*, vol. 18, no. 1, pp. 1–24, 2013. Article (CrossRef Link).
[19] M. Shanker, J. Jaya, and K. Thanushkodi, "An Effective Approach to Software Cost Estimation Based on Soft Computing Techniques," *International Arab Journal of Information Technology (IAJIT)*, vol. 12, 2015.
[20] P. Phannachitta, J. Keung, A. Monden, and K. Matsumoto, "A stability assessment of solution adaptation techniques for analogy-based software effort estimation," *Empirical Software Engineering*, vol. 22, no. 1, pp. 474–504, 2017. Article (CrossRef Link).
[21] A. Idri, M. Hosni, and A. Abran, "Improved estimation of software development effort using Classical and Fuzzy Analogy ensembles," *Appl Soft Comput*, vol. 49, pp. 990–1019, 2016. Article (CrossRef Link).
[22] I. Abnane and A. Idri, "Evaluating fuzzy analogy on incomplete software projects data," in *Proc. of 2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1–8, 2016.
[23] T. R. Benala and R. Bandarupalli, "Least square support vector machine in analogy-based software development effort estimation," in *Proc. of 2016 International Conference on Recent Advances and Innovations in Engineering (ICRAIE)*, pp. 1–6, 2016.
[24] D. Manikavelan and R. Ponnusamy, "Minimizing Analogy Errors with the Help of Fuzzy," *International Journal of Applied Engineering Research*, vol. 13, no. 6, pp. 4527–4530, 2018.
[25] J. H. C. Wu and J. W. Keung, "Utilizing cluster quality in hierarchical clustering for analogy-based software effort estimation," in *Proc. of 2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, pp. 1–4, 2017.
[26] P. Phannachitta, "Robust comparison of similarity measures in analogy based software effort estimation," in *Proc. of 2017 11th International Conference on Software, Knowledge, Information Management and Applications (SKIMA)*, pp. 1–7, 2017.
[27] V. Nguyen, T. Huynh, B. Boehm, L. Huang, and T. Truong, "Investigating the use of duration-based windows and estimation by analogy for COCOMO," *Journal of Software: Evolution and Process*, vol. 31, no. 10, p. e2176, 2019. Article (CrossRef Link).
[28] P. Phannachitta, "On an optimal analogy-based software effort estimation," *Inf Softw Technol*, vol. 125, p. 106330, 2020. Article (CrossRef Link).
[29] A. Idri, F. A. Amazal, and A. Abran, "Accuracy comparison of analogy-based software development effort estimation techniques," *International Journal of Intelligent Systems*, vol. 31, no. 2, pp. 128–152, 2016. Article (CrossRef Link).
[30] F. A. Amazal and A. Idri, "Estimating software development effort using fuzzy clustering-based analogy," *Journal of Software: Evolution and Process*, vol. 33, no. 4, p. e2324, 2021. Article (CrossRef Link).
[31] A. Kaushik, P. Kaur, N. Choudhary, and Priyanka, "Stacking regularization in analogy-based software effort estimation," *Soft Computing*, vol. 26, no. 3, pp. 1197–1216, Feb. 2022. Article (CrossRef Link).
[32] M. Dashti, T. J. Gandomani, D. H. Adeh, H. Zulzalil, and A. B. M. Sultan, "LEMABE: a novel framework to improve analogy-based software cost estimation using learnable evolution model," *PeerJ Comput Sci*, vol. 8, p. e800, 2022. Article (CrossRef Link).

[33] M. Azzeh, Y. Elsheikh, and M. Alseid, "An optimized analogy-based project effort estimation," *arXiv preprint arXiv:1703.04563*, 2017. Article (CrossRef Link).
[34] S. Ezghari and A. Zahi, "Uncertainty management in software effort estimation using a consistent fuzzy analogy-based method," *Applied Soft Computing*, vol. 67, pp. 540–557, 2018. Article (CrossRef Link).
[35] S. D. N. S. S. B. Rumjaun, K. A. Gutteea, and L. Nagowah, "Effortest-an enhanced software effort estimation by analogy method," *ADBU Journal of Engineering Technology*, vol. 5, no. 2, 2016.
[36] F. Fellir, K. Nafil, R. Touahni, and L. Chung, "Improving case based software effort estimation using a multi-criteria decision technique," in *Proc. of Computer Science On-line Conference*, pp. 438–451, 20108.
[37] F. A. Amazal, A. Idri, and A. Abran, "Software development effort estimation using classical and fuzzy analogy: a cross-validation comparative study," *Int J Comput Intell Appl*, vol. 13, no. 03, p. 1450013, 2014.
[38] L. H. Putnam, "Estimating software cost," *Datamation*, pp. 171–178, 1979.
[39] M. Usman, J. Börstler, and K. Petersen, "An effort estimation taxonomy for agile software development," *International Journal of Software Engineering and Knowledge Engineering*, vol. 27, no. 04, pp. 641–674, 2017.
[40] S. K. T. Ziauddin and S. Zia, "An effort estimation model for agile software development," *Advances in computer science and its applications (ACSA)*, vol. 2, no. 1, pp. 314–324, 2012.

Mohammad Ayub Latif is a PhD candidate at Karachi Institute of Economics and Technology (KIET), he is also associated as a full-time faculty member at KIET with the College of Computing and Information Sciences. He has many publications in international journals and conferences. He has also written a book chapter for a book on Global Software Engineering which is published by IGI Global. His research interest are related to software metrics and measurements where the core focus is on software costing and estimation.

Prof. Dr. Muhammad Khalid Khan is the Associate Dean & Director - Faculty of Computing and Information Sciences at Karachi Institute of Economics and Technology (KIET), Pakistan. He is a seasoned professional having an experience of over two decades in Information System implementation, training, coaching and research. He holds a Post Doctorate in Blockchain and Predictive Analytics from UTP, Malaysia and PhD in Computer Science. He has attended two Master programs, one in Computer Science and the other in Business Administration. In the academic leadership role, Dr. Khalid is responsible for more than 2000 students and 100+ faculty and staff members. He is looking after six Bachelors Programs, two Masters Programs and a PhD Program at KIET. He regularly conducts corporate trainings for various commercial organizations. Dr. Khalid has won many research funding from National and International agencies including UTP Malaysia and NRPU, TDF, NCAI funding from HEC Pakistan. The total award is above 30 Million PKR. Dr. Khalid has published and presented more than 60+ articles in peer reviewed international journals and conferences. His citation count is well above 600. For complete list (https://scholar.google.com.pk/citations?user=6aqBzAgAAAAJ&hl=en) Dr. Khalid's research interest include Blockchain scalability & Trust Mechanisms, Desktop Grids & Decentralized clouds, IoT based Healthcare & Air Quality Monitoring, Business Intelligence & Predictive Analytics, Technology Startups & Emerging Software Development Practices.

Dr. Umema Hani received PhD in Software Engineering from (GSESIT, Hamdard University) in 2019. She is working for industry and academia in a domain of Software Engineering since year 2000. She has done research in areas such as, Process improvement, Quality benefit measurement, Software Estimation models, Productivity measurement, IoT, Secure coding, Size estimation using Empirical and Case study based research.