Improving Estimation Accuracy Prediction of Software Development Effort: A Proposed Ensemble Model

Yasir Mahmood*
Advanced Informatics Department, Razak Faculty of Technology and Informatics
Universiti Teknologi Malaysia
Kuala Lumpur, Malaysia
mahmood.yasir@graduate.utm.my
0000-0003-1578-5229

Department of Software Engineering, Faculty of Information Technology
The University of Lahore
Lahore, Pakistan
yasir.mahmood@se.uol.edu.pk

Nazri Kama
Advanced Informatics Department, Razak Faculty of Technology and Informatics
Universiti Teknologi Malaysia
Kuala Lumpur, Malaysia
mdnazri@utm.my

Mazlan Ali
Perdana Centre, Razak Faculty of Technology and Informatics
Universiti Teknologi Malaysia
Kuala Lumpur, Malaysia
mazlanali.kl@utm.my

Azri Azmi
Advanced Informatics Department, Razak Faculty of Technology and Informatics
Universiti Teknologi Malaysia
Kuala Lumpur, Malaysia
azriazmi@utm.my

Abstract— **Software effort estimation is an essential feature of software engineering for effective planning, controlling and delivering successful software projects. The overestimation and underestimation both are the key challenges for future software development. The failure to acknowledge the effort estimation accuracy may lead to customer disappointment, inaccurate estimation and hence, contribute to either poor software development process or project failure. The main aim of this research is to optimize the estimation accuracy prediction of software development effort to support software development firms and practitioners. In this paper, we propose an ensemble software effort estimation model based on Use Case Points (UCP), expert judgment and Case-Based Reasoning (CBR) techniques. This research is conducted through primary (a multi-case involving software companies) study to make an ensemble model. The estimation accuracy prediction of the proposed model will be evaluated by selecting projects from primary studies as case selections in applying a quantitative approach through industrial experts, archival data about estimates and evaluation metrics. The proposed model produced at the end of this research will be used by software development firms and practitioners as an instrument to estimate the effort required to develop new software projects at an earlier stage.**

Keywords— Software development, effort estimation accuracy, ensemble effort estimation, ensemble model

I. INTRODUCTION

Software Effort Estimation (SEE) studies have started since the 1960s and continuous research has been conducted due to numerous claims on attaining accurate estimation results [1]. It is defined as a process of predicting the amount of work and hours needed to develop software. It is usually expressed in man-hours or man-months unit. Today, developing software systems are expensive and difficult. Software engineering presents several ways to quantify a project. One of the most important steps in the software engineering process is to accurately estimate the software engineering process that includes cost, effort and time which has an important role in determining the success or failure of the project. The software development cost and effort estimation are important in the development process and customer requirements. The reports on conducting projects show that there is almost no control over software projects and usually, the scale of the accomplished work is more than what has been estimated before. Therefore, usually, projects terminate later than planned time.

In software engineering, effort estimation would enable managers to estimate, forecast, and accurately quote the requirements for schedule, budget and manpower to successfully complete software projects. The delivery of quality software to end-user within the estimated time and cost remains a major challenge for the software project team. Several studies have emphasized the importance of the role of the software project manager in the project's success or failure. [2]. According to Lehtinen, M [3], failure of a software project indicates recognizable cost, scope, effort, schedule, or quality failure. According to the chaos report (2015) of The Standish Group International, 60% of IT projects were not on their scheduled time and 56% were not on the budget. The International Society of Parametric Analysis (ISPA) studied that inaccurately estimating the staff's skills level, underestimating software size and lack of requirement's understandings are some of the core reasons behind project failures. To date, researchers have therefore introduced different types of SEE techniques. However, most of the techniques were proposed at the beginning of the software development phase based on pre-defined requirements.

Software effort estimation is broadly divided into three main categories: 1) algorithmic, 2) expert estimation and 3) machine learning. In light of these estimation techniques, the experts and practitioners proposed to develop numerous estimation methods for accomplishing high effort estimation accuracy and afterward chosen just a single best method to utilize. However, there is no consensus between the research

978-1-7281-7116-6/20/$31.00 ©2020 IEEE

communities that concludes the best solo method. Recently, new endeavors on ensemble estimation methods have been proposed [4-6]. An ensemble effort estimation technique consists of combinations of more than one single technique to estimate the software development effort of a new project using a combination rule i.e. mean, median, Inverse Rank Weighted Mean-IRWM, etc. The estimation of each base model is combined that produced the estimation of an ensemble.

In this paper, an ensemble model is proposed that incorporated Use Case Points (UCP), expert judgment and Case-Based Reasoning (CBR) techniques to improve the estimation accuracy prediction of software development effort. The rest of this paper is organized as follows: Section 2 presents the problem background. Section 3 provides brief literature on the main software engineering areas investigated in this research. The research methodology is described in Section 4. Section 5 presents the proposed model development. The data collection, analysis, and evaluation are presented in section 6. The conclusion and future work are described in Section 7.

II. PROBLEM BACKGROUND

Any software project's success depends primarily on its accuracy in estimating effort. To date, a lot of research has been conducted to estimate the accuracy of software effort using distinctive techniques. In any case, researchers and specialists are striving to recognize which estimation technique gives increasingly accurate outcomes on the given datasets and the other applicable attributes. The Project Management Institute (PMI) conducted a survey in 2017, investigated that 69% of software successfully achieved the project's original goals and business priorities, 43% were not finished within their initial budgets, 48% were delivered late and 32% failed due to budget lost [7]. Doloi [8], revealed that regardless of management competence and the financial strength of the contractor, accurate cost and effort estimation is the key to avoid cost overrun in projects.

In 2019, an enterprise International Project Management Association (IPMA) conducted a survey of 100 software businesses across a broad section of industries. Their key findings showed that it continues to be difficult for organizations globally to deliver projects that meet all of the objectives across the iron triangle of time, cost and scope, along with achieving stakeholder satisfaction. The survey found that the previous year's incredible 70% of companies suffered at least one project failure. IBM-PMO Consultants conducted a survey of 1,500 change management executives. IBM survey in the project success/failure levels shows that only 40% of projects meet schedule, budget and quality goals, underestimating the complexity of the project is also identified as a challenging factor in 35% of the projects.

III. THE REVIEW OF LITERATURE

Over the past few decades, a lot of work has been performed on various types of effort estimation techniques and a lot of models have been proposed to achieve high effort estimation accuracy [9, 10]. The use of regression analysis; also known as algorithmic estimation is one way to estimate software effort. It uses software size variables such as Line of Code (LOC) and Function Point (FP) as independent variables for regression-based estimation. Sabrjoo, S., et al. [20] compared the COCOMO model using KLOC with COCOMO using Function Point Analysis (FPA). They investigated that the COCOMO model used with FPA has given more accurate results than KLOC using MMRE evaluation measure. Grimstad and Jørgensen [11], compared the accuracy of expert judgment and composite model-based effort estimation techniques. Their primary results indicated that there is a substantial statistical difference between the two techniques. As a result, they strongly suggested that model-based estimation generally will lead to more accurate effort estimation. However, their final results show that the accuracy and effectiveness of the effort estimation techniques are dependent on the goal of their error measurement. Thus, no effort estimation technique that generally could produce accurate results for all analysis goals.

Wu, Li [12], integrated CBR and Particle Swarm Optimization (PSO) methods for software effort estimation. The experimental results indicated that for the three performance variables, the integrated approach combining PSO and CBR has enhanced estimation accuracy at both the training and test stages. Ardiansyah, Mardhia [13], proposed an analogy effort estimation model by modifying Euclidean, Manhattan and Minkowski distance measurements. The best results are obtained with Manhattan distance with a 50% MMRE, 28% MdMRE and PRED (25) at 48%. The mean accuracy of the analogy method is MMRE 49.9%, MdMRE 29.37% and PRED (25) 51.23%.

According to Jørgensen [14], the expert estimation accuracy is better than model-based estimation because the information available with humans can be used flexibly than model-based. The expert judgment can become an ad-hoc activity without using explicit support structure. The experts may forget relevant activities and tasks which are the key reasons for underestimating the development effort [14]. According to Furulund and Molkken-stvold [15], the use of checklists improves effort estimation accuracy. They also found that the projects estimated with checklists were more accurate than the projects estimated without checklists. Jorgensen and Moløkken-Østvold [16], proposed an initial estimation checklist consisting of four phases and twelve activities (preparation phase, estimation phase, the application phase, and learning phase) from several sources. Consequently, experts need to distinguish the circumstances, when to utilize expert estimation and formal/structured model.

The data mining studies reported that ensemble methods provide accurate results compared to single method. This has inspired the researchers to use ensemble methods in various software engineering sub-fields [17]. The basic idea behind using Ensemble Effort Estimation (EEE) is that every single technique has its strengths and weaknesses, we can minimize the limitations by integrating techniques via EEE, which may lead to more accurate estimates. An ensemble technique uses a combination rule i.e. mean, median, Inverse Rank Weighted Mean-IRWM, etc. to make an ensemble.

The researchers have conducted various empirical studies to evaluate ensemble effort estimation techniques and some of them have focused on dealing only with

homogeneous ensembles, heterogeneous ensembles, or both types of EEE techniques [18]. The literature stated that two conditions should be met by the base methods that make up the ensemble: high accuracy and heterogeneity to achieve a high estimation accuracy. In other words, the base methods ought to be diverse and as accurate as could be expected under the circumstances. Every base technique may, therefore, cancel estimation errors made using other base methods.

IV. RESEARCH METHODOLOGY

The engineering methodology is conducted by analyzing existing solutions; to know-how to develop a better solution and test it to verify the hypotheses. The main emphasis is concerned with the design and construction of a more effective solution than the existing solutions. The engineering method has four steps shown in Fig.1. In the first step, the existing models, methods, techniques or frameworks of effort estimation are investigated. In step two, after a detailed analysis of the existing solution, an ensemble model is proposed to optimize the accuracy prediction of software development effort. In the third step, an ensemble model is developed using UCP, expert judgment and CBR techniques. The model development stage will be discussed in section 5. In the last step, the proposed model will be evaluated and in case of failure to reach the objectives of this study, these four steps shall be repeated to find a better solution for the problems in this research.

Fig. 1. Engineering research method process

To design the research plan, this research chosen the guidelines proposed by Wohlin and Aurum [19]. The research structure split into three phases with eight decision points shown in Fig. 2. In addition, different methods can be used to execute each decision point.

As a result, this work used Wohlin and Aurum's guidelines and mapped the research decision-making process of the research design with the research structure shown in Fig. 3.

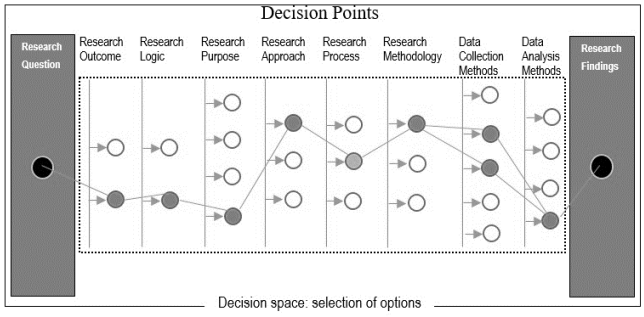

Fig. 2. Research Structure. Wohlin and Aurum [19]

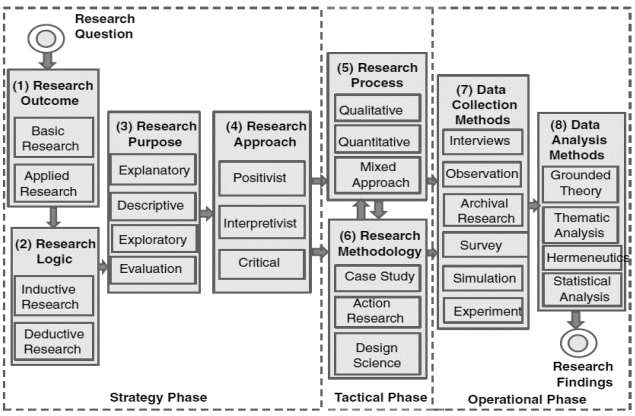

Fig. 3. Research decision-making process

V. PROPOSED MODEL DEVELOPMENT

In the proposed model, three effort estimation models are used as a base model to create an ensemble shown in Fig. 4. The Use Case Points-S_1, Expert Judgement-S_2 and Case-Based Reasoning-S_3 models are parametric, non-algorithmic and machine learning in nature respectively. The use case points are a widely used method for measuring software size and estimating effort in an object-oriented software environment. The use case diagrams are converted into size metrics that classify actors and use cases as simple, average and complex. The calculation of Unadjusted Use Case Points (UUCP) is based on the sum of Unadjusted Actor Weight (UAW) and Unadjusted Use Case Weight (UUCW). Finally, UUCP is adjusted by using Technical Complexity Factor (TCF) and Environmental Factor (EF). TCF affects the performance of the project, estimated from 13 technical factors (T1-T13). Similarly, EF is determined from 8 environmental factors (E1-E8) which influence productivity. The adjusted UCP size will be obtained. The productivity factor is used when estimating the effort involved in a new software project. At the end of this technique, the effort Y_1 will be estimated. In the expert judgment, the involved experts rely on their experience to arrive at effort estimates by considering many relevant factors that they think may contribute to the development effort. Checklists have been used to support experts in consistently using the same set of factors, activities and tasks during effort estimation. It also provides a low-cost mechanism for documenting the important estimation experience, which can be used in the future to improve estimation accuracy. In this research, the combination of checklist proposed by Usman, Petersen [20] and UCP size as input from the UCP technique are used to estimate software effort.

A Case-Based Reasoning (CBR) is the use of an analogy approach to machine learning and is the most investigated method of estimating the effort based on machine learning. CBR's first implementation was proposed in 1997 in software effort estimation [21]. Due to its conceptual simplicity and quantitative reliability, CBR is the most commonly used approach in the software industry. Many research findings suggest that the CBR approach is ideal for estimating effort in software development [22].

The CBR approach identifies one or more existing projects that are similar to the target project and then extract

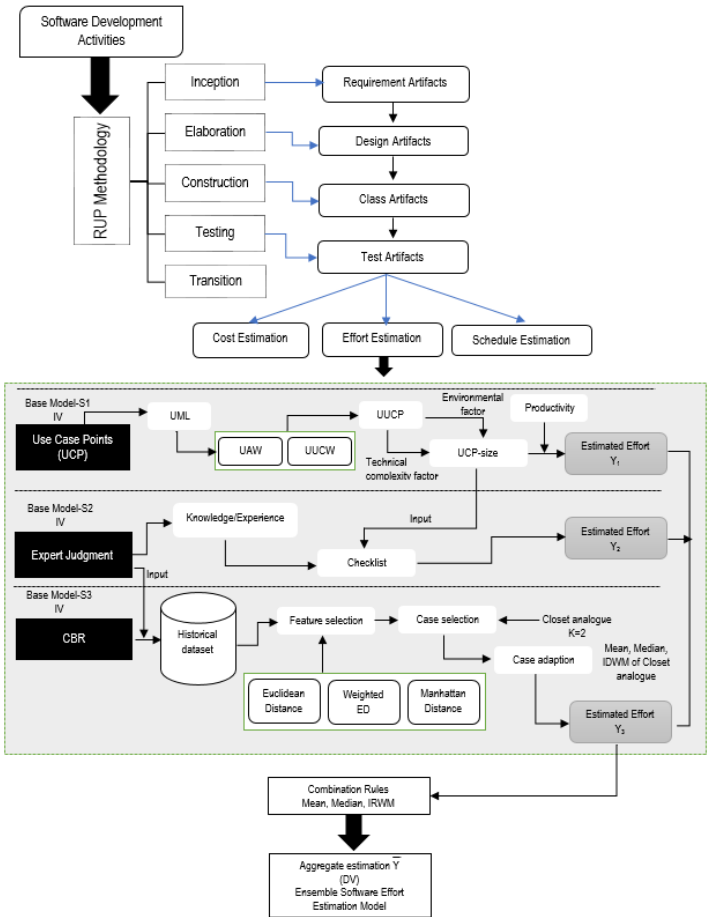

Fig. 4. Proposed ensemble model

the estimation of effort from those projects. The CBR process is shown in Fig. 5. There are three main parameters in the CBR technique, a) the similarity measure, b) the number of most similar projects, c) project adaptation.

The measurement of similarity accounts for the similarity degree between cases. To date in the literature, numerous similarity measures have been proposed [23]. The three most frequently being used are the unweighted Euclidean distance, the weighted Euclidean distance, and the Manhattan distance. The number of analogies is the number of the most similar cases that are used to estimate the effort. A small number of most similar analogues should be considered for small datasets. This approach is used in numerous studies in Web and software engineering. For example, the estimation of effort for a new project can be obtained by using the closest case or analogue (k = 1). Two or three closest analogues are also used in other studies [24]; however, the majority of studies used no more than three closest analogues. Once a similarity measure has been used and chosen, the next step is to decide how to produce the effort estimate of project P_{target}. To illustrate just a few, The selection of analogy adaption techniques proposed in the literature are nearest neighbor, the mean of the closest analogues, the median of the closest analogues, the inverse distance weighted mean, and inverse rank weighted mean [25]. The nearest neighbor uses the same effort for its closest analogue as the estimation effort for P_{target}. The mean of the closest analogues uses as the average of its closest k analogues, when k > 1. Finally, the inverse rank weighted mean allows higher-ranked analogues to influence the outcome more than lower ones. For example, if we use three analogues, weight 3 would be the closest analogue (CA), weight 2 would be the second closest analogue (SC), and weight 1 would be the third closest analogue (LA).

In the proposed model, the UCP, expert judgment and CBR are Independent variables (IVs) whereas effort a dependent variable (DV). The estimation results of UCP (Y_1), expert judgment (Y_2) and CBR (Y_3) are ensemble using combination rules (mean, median and inverse rank weighted mean). Finally, the effort is calculated based on these three base models, hence an ensemble model is developed.

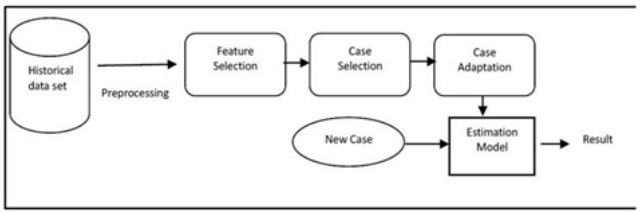

Fig. 5. Case-Based Reasoning (CBR) process

VI. DATA COLLECTION, ANALYSIS AND EVALUATION

To carry out this research, we have chosen a case study methodology. Fig. 6 shows the five major steps for conducting a case study [26].

STEP-1: The detailed structure of steps of conducting the data collection and evaluation phase is shown in Fig. 7. In the first step, a suitable case study will be designed which should be small and typical software development projects that are developed by average developers. The relevant and appropriate case studies will be selected based on the types of developed projects. For this purpose, three software projects (SP_1, SP_2 and SP_3) as case selection from two

software houses (SH_1 and SH_2) will be identified. This research will use a purposive sampling method to select software projects as case selection. It uses an expert's decision to pick cases, or it selects cases that have a particular objective in mind. The subjects and case selections are based on the following criteria:

- The limited number of software projects that either adopted conventional or agile methodologies.
- Software projects which are fully developed in any programming language.
- Software experts (minimum of five-year experience)

STEP-2: The second step is to prepare a case study for data collection. We will use a quantitative (e.g., archival data about estimates) methodology in the case studies.

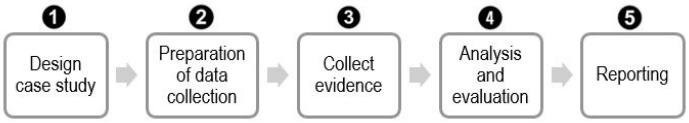

Fig. 6. Steps in conducting a case study

STEP-3: The third step is to collect the evidence from case studies. The usage of appropriate fieldwork is essential for accurate effort estimation. The SRS documents, progress report, software design document, project plans, time reports, effort estimation spreadsheets, etc. will be collected.

The experience of the expert in the software house will be collected and analyzed through proper form fill-in procedure to ensure that the appropriate experts are available in the software to estimate effort. The experience and skills of the experts will be utilized to collect the numbers of similar historical projects for the CBR method. The evidence collected will be used for further evaluation and results.

STEP-4: Step four is divided into two stages. STAGE-1 Computation of effort estimation: The effort and the size of the selected cases will be calculated using the UCP method. Besides, for expert judgment, the checklist will be distributed among the experts required to complete the projects. The size of the software projects identified from the UCP method will be used by the experts for estimation. The experts will use their experiences from the past finished projects to estimate the effort. Similarly, in the CBR technique, the case selection from the repository will be selected by experts and similarity measures will be calculated using Euclidean distance, Weighted Euclidean distance and Manhattan distance followed by case selection and adaption to make an ensemble model using combination rules. Hence, the estimated effort will be computed. The fifth step of reporting will be carried out after the evaluation phase. In this research, we want to improve the estimation accuracy prediction of the proposed model and not the initial effort estimation; therefore, the actual effort and its calculations are considered as the initial effort estimation results.

STAGE-2: The actual estimation and estimation generated by the proposed model shall be compared for evaluation. The evaluation method used for this model is the case study with several controlled experiences. The stages of case study preparation, data collection and calculation of estimated effort using an ensemble model have been described in section 6. The evaluation will be carried out using two evaluation aspects:

FIRST ASPECT: To evaluate the accuracy improvement of the proposed model, the existing models (UCP, expert judgment and CBR) shall be compared individually with the proposed ensemble model. The results generated by these models will be statistically analyzed using evaluation metrics.

SECOND ASPECT: The six most widely used evaluation metrics such as Relative Error (RE), Magnitude of Relative Error (MRE), Mean Magnitude of Relative Error (MMRE), Median Magnitude of Relative Error (MdMRE), Magnitude of Error Relative (MER) and PRED (25) will be used to show the model predictive reliability and the applicability of the proposed ensemble model. The results will be statistically analyzed using mean, median, standard deviation, Cronbach α, t-test, and Pearson correlation.

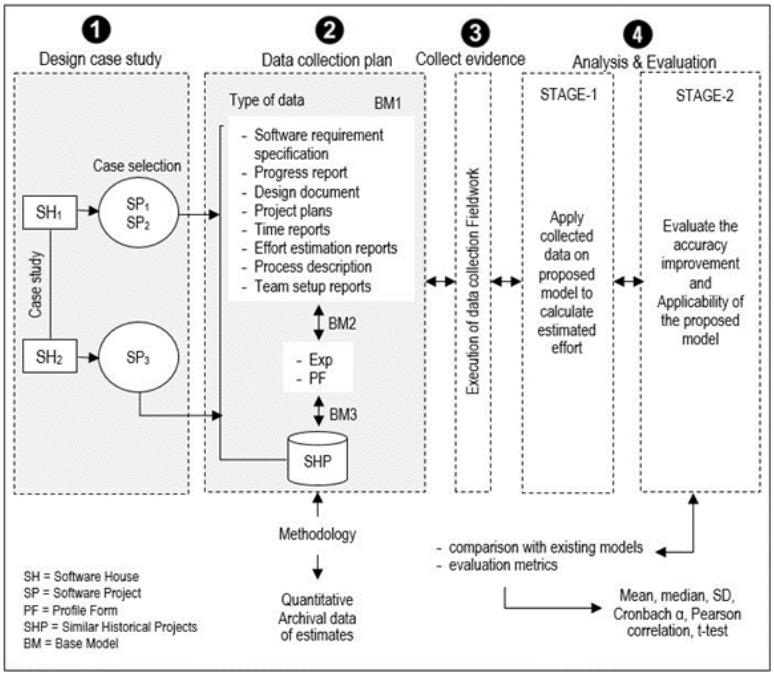

Fig. 7. Steps of conducting a case study, data collection and evaluation

VII. CONCLUSION AND FUTURE WORK

Estimation of software effort is a significant factor for effective software development planning, budgeting and scheduling. The effort estimation has been divided into three main categories algorithmic, non-algorithmic and machine learning. Lots of models and techniques have been developed by the research community to achieve high effort estimation accuracy. It is investigated that the accuracy improvement using the ensemble technique gaining researchers' attention towards further exploring this technique for improving effort estimation results. This is due to the fact that each solo estimation technique has merits and demerits which leads to somehow inaccurate estimation results. This research proposed an ensemble model to improve estimation accuracy prediction of software development effort. The proposed model is incorporated with UCP, expert judgment and CBR techniques to make an ensemble. This research is a part of our ongoing research on improving software development effort prediction of estimation accuracy. In the future, the prediction of estimation accuracy of the proposed model will be evaluated by selecting projects from primary studies as case selections in applying a quantitative approach through industrial experts, archival data about estimates and evaluation metrics as described in section 6 of this paper.

ACKNOWLEDGMENT

This research is fully-supported by Universiti Teknologi Malaysia under the UTM Fundamental Research Grant (UTMFR) with Cost Center No Q.K130000.2556.21H14.

YASIR MAHMOOD
https://orcid.org/0000-0003-1578-5229

REFERENCES

1. Bardsiri, V.K., et al., *LMES: A localized multi-estimator model to estimate software development effort.* Engineering Applications of Artificial Intelligence, 2013. **26**(10): p. 2624-2640.
2. Medina, A. and A.J. Francis, *What are the characteristics that software development project team members associate with a good project manager?* Project Management Journal, 2015. **46**(5): p. 81-93.
3. Lehtinen, T.O.A., et al., *Perceived causes of software project failures - An analysis of their relationships.* Inf. Softw. Technol., 2014. **56**(6): p. 623-643.
4. Kocaguneli, E., et al., *Exploiting the Essential Assumptions of Analogy-Based Effort Estimation.* IEEE Transactions on Software Engineering, 2012. **38**(2): p. 425-438.
5. Minku, L.L. and X. Yao, *Ensembles and locality: Insight on improving software effort estimation.* Information and Software Technology, 2013. **55**(8): p. 1512-1528.
6. Pai, D.R., K.S. McFall, and G.H. Subramanian, *Software Effort Estimation Using a Neural Network Ensemble.* Journal of Computer Information Systems, 2013. **53**(4): p. 49-58.
7. PMI's., *P. M. Institute. Success rates rise: Transforming the high cost of low performance. PMI's PULSE of the PROFESSION*. 2017.
8. Doloi, H., *Cost Overruns and Failure in Project Management: Understanding the Roles of Key Stakeholders in Construction Projects.* Journal of Construction Engineering and Management, 2013. **139**(3): p. 267-279.
9. Yurdakurban, V. and N. ErdoĞan. *S-7 Comparison of machine learning methods for software project effort estimation*. in *2018 26th Signal Processing and Communications Applications Conference (SIU)*. 2018.
10. Sharma, P. and J. Singh. *Systematic Literature Review on Software Effort Estimation Using Machine Learning Approaches*. in *2017 International Conference on Next Generation Computing and Information Systems (ICNGCIS)*. 2017.
11. Grimstad, S. and M. Jørgensen, *S-31 Inconsistency of expert judgment-based estimates of software development effort.* Journal of Systems and Software, 2007. **80**(11): p. 1770-1777.
12. Wu, D., J. Li, and C. Bao, *Case-based reasoning with optimized weight derived by particle swarm optimization for software effort estimation.* Soft Computing, 2018. **22**(16): p. 5299-5310.
13. Ardiansyah, M.M. Mardhia, and S. Handayaningsih, *Analogy-based model for software project effort estimation.* International Journal of Advances in Intelligent Informatics, 2018. **4**(3): p. 251-260.
14. Jørgensen, M., *S-30 A review of studies on expert estimation of software development effort.* Journal of Systems and Software, 2004. **70**(1-2): p. 37-60.
15. Furulund, K.M. and K. Molkken-stvold. *Increasing Software Effort Estimation Accuracy Using Experience Data, Estimation Models and Checklists*. in *Seventh International Conference on Quality Software (QSIC 2007)*. 2007.
16. Jorgensen, M. and K. Moløkken-Østvold, *A preliminary checklist for software cost management*. 2003. 134-140.
17. Elish, M.O., T. Helmy, and M.I. Hussain, *Empirical study of homogeneous and heterogeneous ensemble models for software development effort estimation.* Mathematical Problems in Engineering, 2013. **2013**.
18. Kocaguneli, E., T. Menzies, and J.W. Keung, *On the value of ensemble effort estimation.* IEEE Transactions on Software Engineering, 2012. **38**(6): p. 1403-1416.
19. Wohlin, C. and A. Aurum, *Towards a decision-making structure for selecting a research design in empirical software engineering.* Empirical Software Engineering, 2015. **20**(6): p. 1427-1455.
20. Usman, M., et al., *S-26 Developing and using checklists to improve software effort estimation: A multi-case study.* Journal of Systems and Software, 2018. **146**: p. 286-309.
21. Shepperd, M.J. and C. Schofield, *Estimating Software Project Effort Using Analogies.* IEEE Trans. Software Eng., 1997. **23**: p. 736-743.
22. Huang, S.-J., N.-H. Chiu, and L.-W. Chen, *Integration of the grey relational analysis with genetic algorithm for software effort estimation.* European Journal of Operational Research, 2008. **188**(3): p. 898-909.
23. Mendes, E., N. Mosley, and S. Counsell, *Web Effort Estimation*, in *Web Engineering*, E. Mendes and N. Mosley, Editors. 2006, Springer Berlin Heidelberg: Berlin, Heidelberg. p. 29-73.
24. Mendes, E., S. Counsell, and N. Mosley, *Web hypermedia cost estimation: further assessment and comparison of cost estimation modelling techniques.* New Rev. Hypermedia Multimedia, 2003. **8**(1): p. 199-229.
25. Idri, A., M. Hosni, and A. Abran, *Improved estimation of software development effort using Classical and Fuzzy Analogy ensembles.* Applied Soft Computing, 2016. **49**: p. 990-1019.
26. Runeson, P. and M. H, *Guidelines for conducting and reporting case study research in software engineering.* Empirical Softw. Engg., 2009. **14**(2): p. 131-164.