Smart Computing and Systems Engineering, 2020
Department of Industrial Management, Faculty of Science, University of Kelaniya, Sri Lanka

Paper No: SE-08 **Systems Engineering**

Extending use-case point-based software effort estimation for Open Source freelance software development

Dharshitha Srimal Senevirathne*
Department of Industrial Management
Faculty of Science, University of Kelaniya, Sri Lanka
senevira_im14045@stu.kln.ac.lk

Thareendhra Keerthi Wijayasiriwardhane
Department of Industrial Management
Faculty of Science, University of Kelaniya, Sri Lanka
thareen@kln.ac.lk

***Abstract*: Accurately predicting the software development effort is very crucial when delivering the software systems on time, within the budget and with the required functionality. Overestimation of the software development effort can lead to losing the projects whereas underestimation can cause budget and schedule overruns. The development effort of a software project depends on various factors and these effort factors associated with the freelance software development are different from those of traditional software development. Software development companies employ various proprietary tools in their projects for their planning, development, testing, etc. However, freelance software developers functioning under tight budgetary constraints are not in a position to afford them. As a result, they tend to use free and open-source tools for their software developments. There are various types of software effort estimation models proposed, published and practiced in the industry. However, there is no such software effort estimation model specifically proposed to estimate the effort of freelance software development. The main objective of this paper is to extend Use Case Point-based software effort estimation for the open-source freelance software development. Initially, details of open source software projects were collected from several freelance software developers. Based on the use case diagrams, Use Case Points counts are then calculated for each project. Taking other effort drivers associated with open source freelance software development also into account, we then estimate the effort of each software development. Our aim is to explore the viability of using Use Case Points as the main effort driver in estimating the effort of open source freelance software development.**

Keywords: *Freelance software developers, Open source software development, Software effort estimation, Use Case points*

I. INTRODUCTION

Accurately predicting the software development effort of freelance software developers is very crucial when delivering the software systems on time, within the budget and with the required functionality. Overestimation of the software development effort can lead to losing the projects, whereas underestimation can cause for budget and schedule overruns. Since this research is based on the software effort estimation (SEE) of open source (OS) the freelance software development, it is really important for the freelance software developers to make accurate predictions of effort the software projects. There are no SEE models specifically proposed to estimate the effort of OS freelance software development.

A. Freelance Software Development

With the rapid growth of Information Technology (IT), the self-employment rate of the IT industry has started to increase. Most organizations and people tend to get away from the traditional methods of handling businesses. Since the businesses move towards technology-based solutions and systems, hiring a freelancer to come up with better solutions has become a more cost-effective option. It is really important for the freelance developers to properly estimate the effort of the software projects not only to land the projects but also to deliver the projects within the time and budget. Even though there is a rapid increment in the self-employment and freelancing rate the freelancers find it hard to estimate the effort of the projects accurately. Therefore, it is important for freelance developers to identify the factors that affect the effort of an Open Source Software (OSS) project. These factors are known as effort drivers. The effort drivers associated with freelance software development are different from those of traditional software development. In a software company, a software project is performed by a team, which consists of software developers, business analysts, a tech lead, a team lead, etc. However, freelancers tend to work alone or in very small groups.

B. Open Source Software Development

OSS developers are usually expected to use their own hardware and software tools. Similarly, OSS developers are typically not owned or controlled by any organization to monitor and manage their software development process [1]. Software development companies use various proprietary tools in their software projects in development, testing, planning, etc. phases. However, freelance software developers functioning under tight budgetary constraints are not in a position to afford them. As a result, they tend to use, free and OS tools for their software developments.

C. Use Case Points based model

There are various types of the SEE models proposed, published and practiced in the industry. Many of these models use software size as the main effort driver for their estimations. Among the various types of software size measures proposed, Lines of Code (LOC) is the most widely used measure in sizing software products. However, the LOC is only available in the latter stages of the development of a software product. Therefore, as an alternative, Function Points (FP), a measure based on the functionality of a software product has been introduced to quantify the software size in the early stages of software development. However, the service of trained experts is required to get the FP count of software system accurately and thereby the freelance developers are not in a position to use FP for sizing their software products. Most of the modern software developments are carried out using Object-Oriented System Analysis and Design (OOSAD) methodology. In the early stages of the software development life cycle, OOSAD requires to draw use case diagrams to identify the requirements and the functionality of the system. As a result, an extension of FP, based on use cases called Use Case Points (UCP) has been introduced. Therefore, in this research, we discuss the viability of using UCP as the software sizing measure of each software project.

D. Software Effort Estimation

Among the various SEE methods, the analogy is considered as the most commonly used method. Analogy based estimation has shown better evaluation results compared to other machine learning and non-machine learning methods [2]. Since analogy based SEE models are able to learn from previous experiences analogy compares the effort drivers of the proposed software project to previous software project data to estimate the effort of most similar projects.

The goal of this research is to extend a SEE model to accurately predict the effort of OSS projects developed by freelance software developers. In this research, we look into OSS projects of freelance software developers. First, we identify the effort drivers of freelance OS SEE. Since UCP is selected as the software sizing measure of this research, we examine the use case diagrams of each software project. Based on the use case diagrams, the UCP count of each software project is calculated. The identified effort drivers with UCP count is applied to an analogy based SEE model to estimate the effort of each software development. Finally, the validation of the SEE model is done using n-fold cross-validation.

The structure of this paper is as follows: Section 2 presents an overview of related work. Section 3 describes the methodology that this research is conducted. Section 4 presents and discusses the results. Finally, in Section 5 we present the conclusions and future work.

II. RELATED WORK

There are different types of SEE in the industry by the practitioners. COCOMO, COCOMO II, WALSTON-FELIX MODELS, and SLIM are some of them. They use LOC as their main effort driver for their estimations. Since LOC is only available in the latter stages of software development, FP is used as an alternative. FP is a software sizing measure based on the functionality of software projects. It has been introduced to quantify the software size in the early stages of software development [3]. FP was originally proposed for the procedural systems, but it has now been extended to many other software development paradigms such as Object-Oriented systems [4], embedded systems and real-time systems [5], [6]. However, there is an international standard for counting FP and the service of trained experts is required to get the FP count of software systems and thereby the freelance developers functioning under tight budgetary constraints are not in a position to use FP for sizing their software products [7]. Nowadays, most of the modern software development projects are carried out based on OOSAD methodology. In the early stages of the software development life cycle, OOSAD methodology uses use case diagrams to identify the requirements of the system. As a result, an extension of FP based on use cases called UCP has been introduced by Gustav Karner in 1993 and UCP is evolved from FP [3]. UCP requires the use case diagrams and use case descriptions of the software projects that we consider to calculate the effort [8].

A. Estimation by analogy

Effort estimation by analogy has become more popular within the software development research community because of its higher performance in prediction when different data types are used. The concept of this method has been simplified such that the effort of a new project can be estimated by reusing efforts about similar, already documented projects in a dataset, wherein a first step one has to identify similar projects which contain the useful predictions [9]. The predictive performance of analogy based SEE relies heavily on the selection of two interrelated parameters: number of nearest analogies and adjustment strategy [8]. There are many analogy-based SEE models that have been published by many researchers. However, ESTOR, ACE, and ANGEL are the most commonly used models in predicting effort of software development.

1) Estor: ESTOR is an early implementation of an analogy-based tool to estimate software project effort [10]. It was developed by Mukhopadhyay et al. in 1992 [11] in order to evaluate the feasibility of case-based reasoning in SEE. However, in accordance with the concept of case-based reasoning, ESTOR uses the case-based reasoning and the three best analogies to compute effort based on the Inverse Rank Weighted Mean (IRWM) [12].

2) ACE: ACE (Algorithmic Cost Estimator) was developed by Emilie Mendes et al. in the late 90s and it focuses on exploring the benefits of analogy-based estimation. It calculates the difference between the target project and each calculated project in the database. ACE principles involve the use of similarity functions which should be defined to be able to compute the similarity distance of each analogy with respect to the target project. Also, the similarity function helps in the ranking of analogies in terms of the most similar and least similar [10], [12].

3) Angel: Analogy based effort estimation methods generate new predictions based on the assumption that similar projects with respect to features description have similar efforts. The authors of [13] proposed a tool called ANGEL to predict the effort of a project by using the analogies and estimates provided by the completed projects. ANGEL uses historical effort data and the size of completed projects to predict the estimation of the new project [14]. There are many variables that affect the effort estimation. Not all the variables selected will be helpful when finding good analogies. Some variables may create noise. There, the authors [13] have automated the process and provided an environment in which data can be processed, analogies found and estimates produced. This supports the collecting, storing and identifying the most similar projects in order to estimate the effort for a new project. ANGEL is based upon the minimization of Euclidean distance in multi-dimensional space.

B. *Model comparison*

ACE is known to provide a lower degree of accuracy [11]. ESTOR and ANGEL are said to use the same principles in order to provide a list of most similar analogies though, the content of the list is not always accurate. Moreover, ANGEL is computationally expensive as compared to the other methods since it saves and computes similarity for all cases. ESTOR requires additional domain knowledge in order to succeed to accurately estimate projects from very different environments. Therefore, in this research, we extend the ANGEL model to estimate the effort of OSS projects developed by freelance software developers.

However, there are some limitations to the ANGEL tool [14].

- The values of the effort factors are standardized on the basis that each effort factor contributes equally to the value of the target effort factor.
- No specific method to handle out of range values.
- No specific method to handle "multiple-exact- match" conditions.
- No specific mechanism to handle "no-match" condition

C. *Performance measures*

There are many performance measures that researchers use in SEE such as MRE (Magnitude of Relative Error), MIBRE (Mean Inverted Balanced Relative Error), PRED(25) (percentage of predictions failing within 25%), MMRE (Mean of Magnitude Relative Error), etc. MRE measures the error ratio between the actual effort and the predicted effort and it can be explained from the equation (1) where Ei is the actual effort and Ei is the predicted effort.

$$MRE = | E_i - E_i^{|} | / E_i \quad (1)$$

However, MMRE is considered as the de facto standard of all the performance measures out of them [15]. The equation (2) presents the mean of all the MRE values.

$$MMRE = (\sum_{i=1}^{n} MRE) / n \quad (2)$$

III. Methodology

The methodology of this research is conducted via four phases. In the first phase of the study, a thorough review of the literature is done in order to identify the effort drivers of freelance OSS development. Next, we collect details about freelance OSS projects from several freelance software developers. The third phase of the study is focused on selecting an appropriate SEE model and extending it from traditional software development to OS, freelance software development. In the fourth phase, the outcome of the extended model is tested and validated.

A. *Identifying Effort Drivers*

In software companies, a software project is handled by a group of people, which consist of software developers, business analysts, a tech lead, a team lead, etc. [16]. However, freelancers tend to work alone or in very small groups. Expert judgment is one of the most popular effort estimation methods used in the software industry. This would be a costly option for a freelancer. Therefore, freelance software developers are incapable of hiring an industry expert. Software companies use various proprietary tools for software testing, coding, managing projects, etc. Freelancers are not in a position to afford them. As a result, they tend to use, free and OS tools for their software developments.

1) UCP Count: In this research, UCP is selected as the software sizing factor of the freelance, OSS projects. First, the UCP count of each project is calculated. Most of the Technology Complexity Factors (TCF) such as special user training facilities required, access for third parties, response or throughput performance objectives, etc. and Environmental Complexity Factors (ECF) such as familiarity with Rational Unified Process, part-time workers, lead analyst capability, etc. are irrelevant and have no effect in freelance OSS development. Due to this reason, TCF and ECF can be omitted [17]. Therefore, Unadjusted Use Case Points (UUCP) is considered as the main effort factor in sizing the OSS products.

According to Karner [3], the actors in a use case model can be categorized as simple, average and complex.

- Simple: Weighting factor 1
- Average: Weighting factor 2
- Complex: Weighting factor 3

The total unadjusted actor weight (UAW) is measured by counting the number of actors in each category multiplying each total by the defined weighting factor and then adding the products

The use cases are also categorized into three categories and they are simple, average and complex [3]. This categorization depends on the number of transactions, including the transactions in alternative flows. However, included and extending use cases are not considered. A simple use case has 3 or fewer transactions; an average use case has 4 to 7 transactions, and a complex use case has more than 7 transactions. A weighting factor is assigned to each use case category:

- Simple: Weighting factor 5
- Average: Weighting factor 10
- Complex: Weighting factor 15.

The unadjusted use case weights (UUCW) are calculated by counting the number of use cases in each category, and then multiplying each category of use case with its weight and adding the products. The UAW is added to the UUCW to get the UUCP.

2) Application Domain: The software industry as such does not have a specific domain. Rather, this industry would provide services or enable other services by applying the right technologies. A software domain is nothing but the subject area in which a particular project belongs to. It can be scientific development, business development, health/ medical industry, embedded systems, defense systems, etc. The demand for each domain can vary. Some domain may have high demand while the demands for other domains is low [18]. Some developers could be more comfortable in particular domains while some are new to those domains. This means some developers may have academic training or substantial work experience in a particular domain and they understand the design, architecture, domain rules, etc.

3) Programming language: The programming language of an OSS project is one of the important aspects to be considered. It should be selected based on the project requirements. Different programming languages are suitable for different application domains. For example, Python is usually used for scientific systems, whereas PHP and Java are used for web developments. The choice of selecting a programming language for developers depends on the required functionalities of the software project [19].

The programming language selected for a given project should be sufficiently expressive to cover the requirements. On the other hand, an incorrect language selection may affect the solution from reaching the expected level. Some languages are expressive thanks to their verbose nature. It allows more expressivity and hence more choices to be able to make use of simple syntax. Effectively, the simple syntax would eventually result in the developer and reader's familiarity with the code quicker than having the reader being exposed to some new complex syntax [20].

4) Developer experience: Freelance software developers land projects through contacts of their previous projects. The experience in freelancing can't be measured with the time because some freelance software developers could have worked with fewer projects for a long duration of time while another set of developers would have worked with many projects in a short period of time. Therefore, the number of projects they have completed is used as the measurement of the freelance developer's experience. In online crowdsourcing platforms or online market places, they get projects based on their ratings and feedbacks. Rating and feedbacks are given by the project owners of previously completed projects. Therefore, having a good experience is a clear advantage to land more projects [18]. If a developer has previous experience it easy to understand the requirements. When developing a particular feature or solving a particular problem it would take less time to come up with a better solution. Generally, specific skills and depth of experience matter a lot when dealing with complex tasks [18]. Experience helps the developers to select a better approach to provide a solution to a requirement.

B. Data Collection

We have considered OSS projects developed in multiple domains, by software development freelancers in Sri Lanka with different levels of experience. The domains include business applications, embedded systems, data science projects, etc. The complexity of these OSS projects is different and the time taken to complete each project is also different.

We have collected the details of 13 OSS projects from 9 freelance software developers. Based on the use case diagrams, the UUCP count is calculated for each project. There can be some inefficiencies in the use case diagrams. These inefficiencies will affect when calculating the UCP of those projects. Therefore, the total effort estimation of those software projects could vary from its real values. We have identified the erroneous diagrams by grouping the projects according to their domain and by comparing each project according to its UCP and the actual effort. Details of the collected OSS projects collected from freelance software developers are shown in Table I.

TABLE I. OPENSOURCE SOFTWARE PROJECTS OF FREELANCE SOFTWARE DEVELOPERS

No	Developer Experience	Language	Domain	UUCP	Effort (Hours)
1	17	Java	BA	247	201
2	19	Java	BA	304	265.5
3	21	Java	BA	258	264
4	25	PHP	BA	231	201
5	19	PHP	BA	216	209.5
6	23	PHP	BA	192	210
7	16	C	ES	177	224.5
8	19	C	ES	168	191
9	29	Python	ES	204	188
10	27	Python	ES	117	171
11	17	Python	SA	71	140
12	23	Python	SA	73	133
13	18	Pythons	SA	95	97.5

BA: Business Application, ES: Embedded systems, SA: Scientific Application

Since data values are on different scales, we standardize them using the equation given in (3) where X is the actual value, X_{min} is the minimum effort of the data set and X_{max} is the maximum effort of the data set.

$$X = (X-X_{min})/(X_{max}-X_{min}) \tag{3}$$

1) Similarity Measure: A similarity measure is used to calculate the similarity between projects based on how close the distance between projects according to the type of each attribute. The are many measurement techniques like Euclidean, Manhattan, and Minkowski that can be used to measure the distances between projects. Angel uses Euclidean distance is measuring the distance (D) between two software project points in the plane with coordinates (x, y) and (x|, y|) is given by equation (4).

$$D = [(x-x^{|})^2 +(y-y^{|})^2]^{1/2} \tag{4}$$

2) Number of Analogies: The chosen analogy is defined by how many similar projects are used to compare to measure the effort of the OS software project. Basically, there are two ways to select analogies. They are fixed and dynamic analogy selection. This research adopts fixed analogy selection and suggests using one closest analogy (K = 1), two closest analogies K = {1, 2}, three closest analogies K = {1, 2, 3}. K can be defined through the following equation where N is the number of data elements. The equation (5) denotes the value of K.

$$K = (N)^{1/2} \tag{5}$$

3) Analogy Adaptation: The effort of the new project is calculated by using certain statistical techniques. There are four types of analogy adaptations that can be applied. They are the Closest Analogy (CA), the mean of closest analogies, the median of closest analogy, and IRWM of closest analogy.

4) Analogy Adaptation Rules: The effort estimation of a new project is done by dividing the actual effort of the old project effort ($Effort_{old}$) with the size of old project ($Size_{old}$) then multiply with the size of new project ($Size_{new}$). Since the sizing factor of this model is UUCP, the equation (6) denoted the formulations of these adaptation rules [21].

$$Effort_{new} = (Effort_{old} / Size_{old}) \times Size_{new} \tag{6}$$

The n-fold Cross-Validation technique is one of the most used approaches by practitioners for model selection and error estimation of classifiers. The dataset is split into k number of subsets. Then, a portion of the split data set is used for testing while the rest of the data are used to learn the model. This is done iteratively [22]. Finally, by comparing the actual effort against the estimated effort, we validate our effort model using n-fold cross-validation. When a specific value for n is selected, it could be used in place of 'n' with the reference to the model, such as n = 3 becoming 3-fold cross-validation. The model is tested with a portion of data while holding the rest of the dataset as the training set. This process is iterated until the remaining groups as a training data set. Data sets are randomly divided into training data and testing data, with a percentage of 92% and 8% respectively. The accuracy is tested by using MMRE.

IV. RESULTS

We have tested the data with four analogy adaptation methods and the most accurate adaptation method is selected by comparing the MMRE of each method.

5) Closest Analogy: The closest analogy focuses on selecting from the closest project, which means the value of K=1. The estimated effort values for the closest analogy of each project with MRE values are shown in Table II.

6) Mean of the Closest Analogy: The mean of closest analogies is the adaptation analogy obtained by calculating the average effort driver from as many as K > 1 selected analogy. The estimated effort values for the mean of the closest analogies with MRE values are shown in Table III.

TABLE II. CLOSEST ANALOGY

Project No	Estimated Effort (hours)	Actual Effort (hours)	MRE
1	215.72	201	0.07323
2	311.07	265.5	0.17164
3	225.32	264	0.14649
4	252.66	201	0.25699
5	236.25	209.5	0.12768
6	167.06	210	0.20445
7	201.23	224.5	0.10364
8	213.08	191	0.11563
9	231.93	188	0.23366
10	213.16	171	0.24658
11	71.37	140	0.44671
12	73.38	133	0.44824
13	172.60	97.5	0.80739

TABLE III. MEAN OF THE CLOSEST ANALOGY

Project No	Estimated Effort (hours)	Actual Effort (hours)	MRE
1	252.74	201	0.25743
2	247.38	265.5	0.06823
3	209.95	264	0.20473
4	224.05	201	0.11467
5	175.77	209.5	0.16099
6	186.22	210	0.11323
7	177.93	224.5	0.20743
8	168.88	191	0.11579
9	371.67	188	0.97697
10	133.02	171	0.22212
11	129.36	140	0.00276
12	132.63	133	0.00275
13	173.08	97.5	0.81238

7) Median of the Closest Analogy: The median of the closest analogy is an adaptation analogy obtained by calculating the median effort driver from as many as K > 2 selected analogies. The estimated effort values for the median of the closest analogies with MRE values are shown in Table IV.

TABLE IV. MEDIAN OF THE CLOSEST ANALOGY

Project No	Estimated Effort (hours)	Actual Effort (hours)	MRE
1	223.01	201	0.10950
2	272.10	265.5	0.02486
3	218.04	264	0.17411
4	253.01	201	0.25877
5	214.30	209.5	0.02290
6	195.98	210	0.06674
7	206.51	224.5	0.080164
8	203.37	191	0.064739
9	410.32	188	1.18257
10	137.56	171	0.19553
11	128.83	140	0.00130
12	134.13	133	0.00851
13	196.78	97.5	1.06054

8) Inverse Rank Weighted Mean of Closest Analogy: IRWM is an adaptation analogy that gives the highest weight in the selected analogy most similar to other analogy. The estimated effort values for IRWM of the closest analogies with MRE values are shown in Table V. In this research the value of K is 3. Therefore, 3 of the closest analogies are selected, the first closest analogy (CA) is given a weight of three, the second closest analogy (SC) is given a weight of two, and the third closest analogy (TC) is given a weight of one [23]. The calculation of IRWM is formulated in equation (7).

$$IRWM= (3CA+ 2SC+ TC)/6 \quad (7)$$

TABLE V. INVERSE RANK WEIGHTED MEAN

Project No	Estimated Effort (hours)	Actual Effort (hours)	MRE
1	225.54	201	0.12207
2	280.97	265.5	0.05827
3	219.12	264	0.17000
4	248.07	201	0.23416
5	215.19	209.5	0.02718
6	184.72	210	0.12039
7	199.98	224.5	0.10920
8	200.86	191	0.05161
9	344.42	188	0.83200
10	162.01	171	0.05259
11	109.77	140	0.14909
12	113.63	133	0.14561
13	184.77	97.5	0.93480

9) MMRE Results: The evaluation results of MMRE obtained by Euclidian distance had the lowest MMRE of 0.23131 with K = 3 using the IRWM. We have calculated the MMRE values for each type of analogy adaptations.

- Closest Analogy: 0.2602
- Mean of the Closest Analogy: 0.25073
- Median of the Closest Analogy: 0.25002
- Inverse Rank Weighted Mean: 0.23131

Closest analogy and the IRWM perform compared to other methods. IRWM adaptation provides the lowest MMRE value therefore it can be selected as the best adaptation method for this research.

V. CONCLUSION

There are no specific SEE models proposed to estimate the effort of OSS projects of freelance software developers. In this research, we have identified the effort drivers that affect the effort of an OSS project developed by freelance software developers. According to the literature, we have identified UUCP, developer experience, programming language, and application domain as the effort drivers of freelance OSS development. Accurately predicting the effort of software projects is very crucial for freelance developers. One of the most important issues of SEE is related to the estimation accuracy of the software projects. The accuracy of the estimated efforts is heavily reliant on the similarity of the historical data of the projects. Therefore, the similarity of the project to be estimated is very essential to improve the accuracy of the estimation. We selected the ANGEL SEE model, an analogy-based effort estimation model and we extended ANGEL to estimate the effort of OS freelance software development projects.

In this paper, we propose four analogy adaptation methods, namely the closest analogy, the mean of the closest analogies, the median of the closest analogies and IRWM. The MMRE values of these four adaptation methods are respectively 0.2602, 0.25073, 0.25002 and 0.23131. Therefore, it is clear that IRWM performs better when adapting the analogies compared to the other 3 analogy adaptation methods. In this research, we have explored the viability of using UCP as the main effort driver and extension of the ANGEL model to estimate the effort of OS freelance software development.

REFERENCES

[1] W.Scacchi., J. Feller, B. Fitzgerald, S. Hissam, and K. Lakhani, “Understanding Free/Open Source Software Development Processes”, *Software Process--Improvement and Practice*, vol. 11, no. 2, pp95-105, March/April 2006.

[2] A. Idri, F. A. Amazal, and A. Abran, “Analogy-based software development effort estimation: A systematic mapping and review*,” Inf. Softw. Technol., vol. 58, pp. 206–230,* 2015.

[3] G. Karner. “Resource Estimation for Objectory Projects”, *Objective Systems SF AB*, 1993.

[4] G. Costagliola., F. Ferrucci, G. Tortora, and G. Vitiello, “Class point: an approach for the size estimation of object-oriented systems”. *IEEE Transactions on Software Engineering, no31(1), pp52 – 74,* 2005.

[5] C. Jones, *Applied Software Measurement: Assuring Productivity and Quality*: McGraw Hill, 1996 [

[6] A. Abran, M. Maya, J. M. Desharnais and D. St-Pierre, "Adapting Function Points to Real-Time Software," *American Programmer*, vol. 10, 1997

[7] M. Barlage, A. Born. and A. Witteloostuijn, “The needs of freelancers and the characteristics of ‘gigs’: Creating beneficial relations between freelancers and their hiring organizations”, *Emerald Open Research,* August 2019.

[8] M. Damodoran and A.N.E. Washington, “Estimation using Use Case Points”, 2002.

[9] M. Shepperd and C. Schofield, “Estimating software project effort using analogies”, *Journal of IEEE Transaction on Software Engineering, vol. 23, pp736–743,* 1997.

[10] J. Keung, “Software Development Cost Estimation Using Analogy: A Review”, *Software Engineering Conference,ASWEC'09. Australian. IEEE,* 2009.

[11] T. Mukhopadhyay, S. Vincinanza, M. J. Pietula., “Estimating the feasibility of a casebased reasoning model for software effort estimation”,*MIS Quarterly, 16:155,* 1992.

[12] E. Mendes, N. Mosley, S. Counsell, “Web effort estimation”,*ICWE'07 Proceedings of the 7th international conference on Web engineering. Pages 90-104,* 2007.

[13] K. Moløkken amd M. Jorgensen, “A review of surveys on software effort estimation”, *Empirical Software Engineering*, 2003. ISESE 2003.

[14] S. Sridhar, “EXTENDED ANGEL: Knowledge-Based Approach For LOC And Effort Estimation For Multimedia Projects In Medical Domain”, *International Journal of Software Engineering & Applications (IJSEA). vol. 2, no. 4, pp97-105*, 2011.

[15] P. Dan, and M. Korte, “Comparative Studies of the Model Evaluation Criterions MMRE and PRED in Software Cost Estimation Research”, *Second International Symposium on Empirical Software Engineering and Measurement (ESEM 2008),* 2008.

[16] H.P. Patra, K. Rajnish, and U.S. Panda, “A New Software Cost Estimation model for Small Software Organizations: An Empirical Approach.” *International Journal of Applied Engineering Research. vol. 10, no. 15, pp. 36076-36082,* 2015.

[17] P. Mohagheghi, B. Anda, and R. Conradi., “Effort estimation of use cases for incremental large-scale software development,” *in 27th International Conference on Software Engineering. IEEE,* 2005.

[18] A. Dubey, K. Abhinav, S. Taneja, G. Virdi, A. Dwarakanath, A. Kass and M.S. Kuriakose, “Dynamics of Software Development Crowdsourcing”, *Proceedings of the 2016 IEEE 11th International Conference onGlobal Software Engineering (ICGSE), Irvine, CA, USA*, 2–5 August 2016.

[19] Z. Liao, B. Zhao, S. Liu, H. Jin., D. He, L. Yang,, Y. Zhang, J. Wu., "A Prediction Model of the Project Life-Span in Open Source Software Ecosystem", *Mobile Networks and Applications*, vol. 24, no. 4, 2019.

[20] Dorn, J., A General Software Readability Model. University of Virginia, Charlottesville, Virginia. 2012.

[21] A. Ardiansyah, M.M. Mardhia and S. Handayaningsih, "Analogy-based model for software project effort estimation*", International Journal of Advances in Intelligent Informatics, vol 4,no. 3,* November 2018.

[22] D. Anguita, L. Ghelardoni, A. Ghio, L. Oneto, and S. Ridella, "The 'K' in K-fold Cross Validation", *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning,* 2012.

[23] Mendes, Cost Estimation Techniques for Web Projects. *IGI Global*, 2008.