Calibrating Function Complexity in Enhancement Project for Improving Function Points Analysis Estimation

Vo Van Hai[1], Ho Le Thi Kim Nhung[1] and Huynh Thai Hoc [1]

[1] Tomas Bata University in Zlin, Faculty of Applied Informatics, Nad Stranemi 4511, Zlin 76001, Czech Republic
{vo_van, lho, huynh_thai}@utb.cz

Abstract. Producing a good software product on time and within budget, the initial software estimation takes a significant role. Reality has shown that most software fails because the initial software estimation is not correct. Many researchers have proposed methods for software estimation. It has been developed since the 70s of the last century, but it is still of great interest until now. We also know that creating a new software product is difficult; it is even more difficult to innovate. In the framework of this paper, we propose an improved method based on the FPA method of IFPUG. We named this proposed model is Calibrating Function Complexity in Enhancement Project (CFCEP). This method is based on the Linear Regression technique to give coefficients of function complexity. The experimental results based on the ISBSG dataset show that the estimation based on this new coefficient gives much better results than using the coefficients of the standard FPA method.

Keywords: Software Measurement, Function Point Analysis, Effort Estimation, Multiple Linear Regression.

1 Introduction

Software estimation is an essential issue in the process of developing a software system [7]. Many projects have not been completed on time or have ultimately failed, primarily because the effort was underestimated or overestimated [8]. For this reason, various software effort estimation models have been developed to manage the budget and schedule of software projects [5]. Accurate effort estimation is essential for project managers and customers [6]. It supports creating an application for bidding, contract negotiation, scheduling, monitoring, and controlling software development.

According to Moløkken [16], only about 30-40% of software projects are completed. The CHAOS team's report shows a partial or ultimately failure rate of software projects, which was 70% in 1994 [17], and this number increased to 83.9% in 2019 [10]. Several studies have been conducted to find out the reasons for the failure of software projects. Galorath et al. [18] indicate that the causes of this failure are inadequate engineering, poor project planning, sudden decisions at the early stages of the

project, and incredibly inaccurate estimates. Some researchers have also pointed out that the leading cause of project failure is incorrect cost estimates [19] [17] [20]. Therefore, software effort estimation methods need to be improved to evaluate different types of projects. One of the possible research directions could be to study methods based on Functional Point Analysis (further only FPA) [2]. An important aspect to be measured in software engineering projects is the functional size of the software [21]. FPA is a standardized method for determining the size of software based on its functional requirements. It is designed to be applicable regardless of programming language and implementation technology. Albrecht [2] recommends FPA to measure the size of the system that processes data from end-users.

The FPA method has been standardized by ISO/IEC 20926:2010 [22]. However, the FPA method is controversially discussed by different researchers regarding its advantages and limitations. Henderson et al. [23] studied the FPA method from a manager and developer perspective based on 13 attributes with three main findings: (1) Source Lines Of Code (further only SLOC) count is less complex than Functional Point (further only FP); (2) developers better understand FP profitability than managers, and (3) differences between managers and developers in values block the communication needed to make informed decisions. Kampstra et al. [24] and Kemerer [25] reported that the FPA method does not produce consistent results when applied by different metrics. Meli [26] showed a mismatch between the established complexity for Base Functional Components (further only BFC) and possible yield estimates.

Many studies have shown that the FPA for BFC scores is incorrect. For example, the same data function and/or the same transaction function can be classified with different DET and RET/FTR combinations with the same complexity. This results in the same function score for data functions and/or transactional functions. It was also found that in some situations, functions with very similar DET and RET/FTR can be classified with different complexity and therefore receive different weights according to the FPA.

Xia et al. [27] suspect that the unadjusted FP weighted value based on research on data processing systems at IBM (locally) cannot reflect software globally. In [28], they repeatedly point out that the classification is ambiguous, and that the original method may not adequately reflect the reality of software complexity in a particular software application. In [29], they demonstrated no clear boundary between the two classifications in FP counting. The authors propose combining three techniques (fuzzy logic, artificial neural network construction, and statistical regression) in a neural fuzzy function point calibration model to solve these problems.

According to Hajri et al. [30], the classification of function types into simple, average, and complex does not reflect the full extent of complexity required to develop a user system. The main innovative idea of this study is to introduce a new weighting system for measuring FP using the Artificial Neural Network (further only ANN) - using the backpropagation technique. In the first step, the original weighting system is used as a baseline to establish new weights. Next, they train one of the most popular techniques in Neural Networks to predict the value of new weights. Then, the new

weights and the original weights are inserted into the FP model. Finally, the size of FP is calculated as a function of the original weights and the new weights.

Ya-Fang et al. [31] reported the IFPUG FPA barely distinguishing the complexity of adjacent components. If this situation occurs in a software program, the estimated results will not correspond to the actual situation. To address this shortcoming, they believe that fuzzy theory should be used to analyze component complexity. He also said that the weights of the BFCs, as set by the IFPUG, are supposed to reflect the functional size of the software, but software today is vastly different from before, so it no longer exists suitable anymore. Agree with Ya-Fang et al. [31] state that this inconsistency in a large number of BFCs, located over boundary regions of defined periods, is even worse. This is because an incorrect classification of the different functions of the system will skew its functional dimensions. Like Xia et al. [19-21], the authors also link fuzzy logic with artificial neural networks. The neural network can learn from the estimated data from the rules and linguistic terms defined by fuzzy logic.

In recent decades, scientists have done a lot of research on the implementation of localized learning using soft computational techniques and statistical techniques. Specifically, Šilhavý et al. [34] proposed a new categorical variable segmentation model based on dataset segmentation to estimate software development effort using a remarkable model trained on a particular data segment. In addition, Prokopová et al. [35] also determined the influence of the VAF factor on the accuracy of the software cost estimation process. Wena et al. [36] showed that the ANN technique is the most widely used and provides the most accurate results along with Support Vector Regression (further only SVR). Although soft computational techniques have shown positive results, there is a significant variation between results reported from different studies.

Since there are many of the above limitations, this study proposes a function complexity weight calibrated based on multiple linear regression for estimating the development effort of enhancement software projects when the IFPUG FPA method is used for baseline estimation. This study will answer the research question of whether the proposed method can significantly improve the prediction of the IFPUG FPA method. The methods are applied to the International Software Benchmarking Standards Group (ISBSG) dataset, which contains data on completed software projects [4]. Unbiased evaluation criteria (eq. 9-11) were used to evaluate the estimation accuracy of the methods.

The remainder of the paper is organized as follows: Section 2 presents the background of the methods used. Section 3 describes the research methodology. Section 4 introduces the results and the discussion, and Section 5 the conclusions and suggestions for future work.

2 Background

2.1 Function Point Analysis

FP was first introduced by A. J. Albrecht [2] and widely accepted with many variants from both academics and practitioners [32]. The study in this area is also known as FPA or Function Size Measurement (FSM). The function points measurement could be classified into FP counting and FP estimation [33].

The FPA assumes that a software program consists of a set of functions. In turn, each function can be a data transaction or transactional function. The data transaction has consisted of Internal Logic Files (ILF) and External Interface Files (EIF), and the transaction function comprises External Input (EI), External Output (EO), or External Inquiry (EQ). In turn, each of these is judged as simple, average, or complex and assigned a weight accordingly (see **Table 1**).

Table 1. Data and transactional function complexity

		Component				
		EI	EO	EQ	EIF	ILF
Functional Complexity	Low	3	4	3	5	7
	Average	4	5	4	7	10
	High	6	7	6	10	15

The following equation can obtain the total unadjusted function points (UFP)

$$UFP = \sum_{i=1}^{n} \sum_{j-1}^{m} (W_{ij} \times S_{ij}) \quad (1)$$

where W_{ij}'s are the complexity weights and S_{ij}'s are the counts for each function component, *n* is the number of types, and *m* is the number of complexity groups.

The Value Adjustment Factor (VAF) should be aimed at before calculating the function points. The VAF is based on the rate of 14 general system characteristics (GSCs - Data Communications, Distributed Data Processing, Performance, Heavily Used Configuration, Transaction Rate, Online Data Entry, End-User Efficiency, Online Update, Complex Processing, Reusability, Installation Ease, Operational Ease, Multiple Sites, Facilitate Change). The GSCs are rated based on their degree of influence or complexity on the 0 – 5 scale. The following table represents the significance of the influence factors rating.

Table 2. The GSCs Factor weights

System Influence	Rating
Not present or no influence	0
Incidental influence	1
Moderate influence	2
Average influence	3
Significant influence	4
Strong influence throughout	5

The VAF count is adjusted as

$$VAF = 0.65 + 0.01 \times \sum_{i=1}^{14} (F_i \times rating) \quad (2)$$

After all, the adjusted function points (AFP) can be reached by using the following Eq. (3)

$$AFP = UFP \times VAF \quad (3)$$

In the previous IFPUG, the function point was used to develop a new project but not enhance an existing project. However, in the latest version (4.3.1), this problem is changed. We can use FP for both Adjusted Development Project Function Point (aDFP) and Adjusted Enhancement Project Functional Size (aEFP).

Use the following formula to calculate aDFP:

$$aDFP = DFP \times VAF \quad (4)$$

where $DFP = AFP$. In aEFP, the following formula was used to calculate the function size:

$$aEFP = [(ADD + CHGA + CFP) * VAFA] + (DEL * VAFB) \quad (5)$$

where:

ADD - the size of the functions being added by the enhancement project.

CHGA - the size of the functions being changed by the enhancement project – as they are / will be after implementation.

CFP - the size of the conversion function.

VAFA - the value adjustment factor of the application after the enhancement project is complete.

DEL - the size of the functions being deleted by the enhancement project.

VAFB is the value adjustment factor of the application before the enhancement project begins.

2.2 The Multiple Linear Regression

In the data analysis, a common task is to study the dependence of a responding variable $Y \in R^1$ on many independent variables $X = [X_1, X_2, \ldots, X_n] \in R^n$. The model given in the form of a mathematical equation $Y = f(X)$ describing each relationship between Y and X is called the regression model. In the case of only one independent variable, it is called Simple Linear Regression. There are two or more independent variables called Multiple Linear Regression (MLR) [14]. The essential model has the following form [15]:

$$Y = \beta_0 + \beta_1 X_1 + .. + \beta_n X_n + \varepsilon \quad (6)$$

where Y is the predicted dependent variable, n is the number of variables, β_0 is an intercept, X_i are the independent variables and $\beta_i \; i \in 1, n$ are called partial regression coefficients, ε is error residual. In this study, the β_0 and ε value was omitted to get the β_i which does not depend on the intercept and the error residual.

3 Research Methodology

3.1 Data Description

In this article, we use the ISBSG dataset version August 2020 R1 [4]. This dataset has been contributed by companies in the field of information technology for many years. In this experiment, we conduct data preprocessing according to the following criteria: First, we only select records with the counting approach of IFPUG (including IFPUG Old and IFPUG 4+). This study is for enhancement, so we choose the development type as an enhancement. Of course, the Data quality rating is selected as A and B. Since the records are related to functions, the data transaction and transactional function must not be empty.

The data to this point is basically complete; however, in some records with enough data, the value of VAF is missing; we proceed to recalculate this value based on the formula

$$VAF = \frac{AFP}{UFP} \tag{7}$$

Finally, due to some errors in data collection, we proceed to remove records whose sum of added, changed, and deleted functions is different from UFP. At this point, we have a dataset with 911 records ready for testing.

We notice that there is a possibility that the data may be noisy because some values of SWE are too far from the mean group, as shown in **Fig. 1**. In this study, we used the interquartile range (IQR) method to determine the outlier. According to this method, suppose that Q1 is the lower quartile, and Q3 is the upper quartile, we have Eq. (10).

$$\text{IQR} = \text{Q3} - \text{Q1} \tag{8}$$

Any value that is greater than $\text{Q3} + 1.5 \times \text{IRQ}$ or less than $\text{Q1} - 1.5 \times \text{IRQ}$ was considered an outlier. The simulation of this process was shown in **Fig. 2**. After removing outliers, our dataset can be seen in Table 3.

Table 3. Statistic characteristics of SWE

	count	mean	std	min	25%	50%	75%	max
Before remove outliers	911	3413.367	5055.922	16	812.0	1930.0	3867.5	61891.0
After remove outliers	637	3003.243	3327.755	26	929	1974	3705	25900

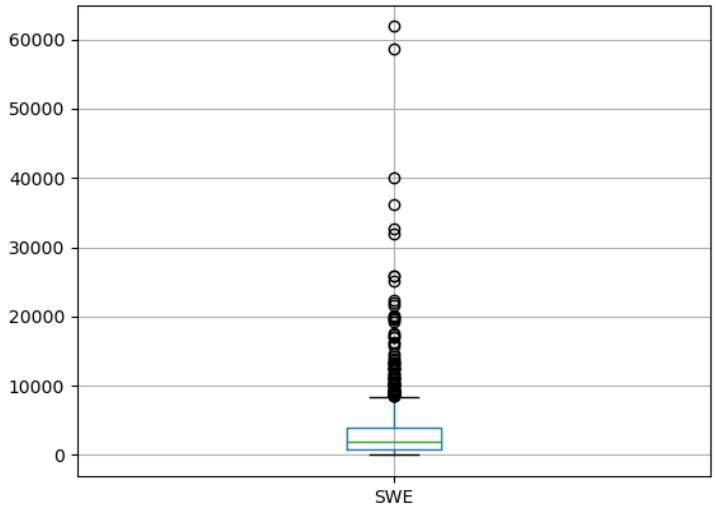

Fig. 1. Boxplot of SWE for dataset

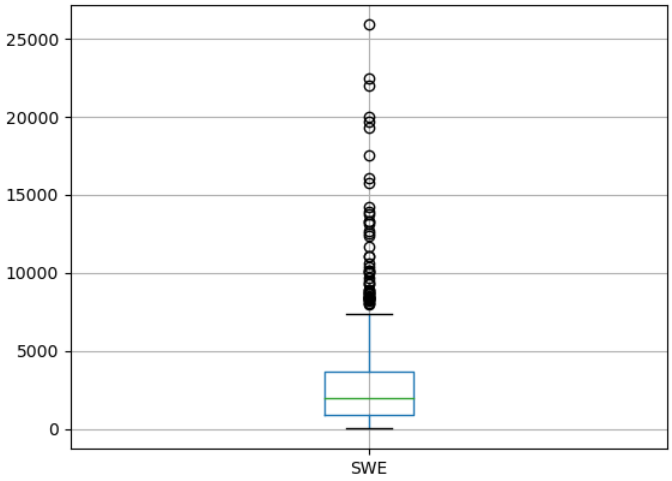

Fig. 2. Boxplot of SWE after removing outliers for dataset

3.2 Evaluation criteria

Regarding the accuracy of the measurement, according to Foss et al. [11], there have been many studies and evaluations on the relevance of the error function that have been proposed and used so far. However, there is no silver bullet to choose the best predictive models among several alternatives. That means that every accuracy metric presents certain advantages over the rest. Kitchenham et al. [12] have shown that the

most crucial factor in making meaningful comparisons between predictive models is determining each error function using actual measures. In this study, to evaluate the accuracy of the proposed model, we use the most popular and widely used measure to evaluate the predictive ability of the comparative models [13].

MAE (Mean Absolute Error). MAE is calculated as:

$$MAE = \frac{\sum_{i=1}^{n} |Effort_{actual} - Effort_{predict}|}{n} \quad (9)$$

where n is the number of test projects, the $Effort_{actual}$ is the actual effort and the $Effort_{predict}$ is the predicted effort. MAE is considered a good choice for measuring symmetric unbiased under or overestimates [9] [10]. A higher (better) predictive performance can be achieved with lower MAE.

MSE (Mean Squared Error) is the mean of the square of the differences between the actual and the predicted efforts.

$$MSE = \frac{\sum_{i=1}^{n} (Effort_{actual} - Effort_{predict})^2}{n} \quad (10)$$

RMSE (Root Mean Square Error) is a frequently used measure that is just the square root of MSE. RMSE can be calculated by using the equation given below

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (Effort_{actual} - Effort_{predict})^2}{n}} \quad (11)$$

3.3 Methodology used

The use of function complexity values shown in Table 1 has many limitations, as mentioned in part 1. In this study, we use MLR to propose changes on these function complexity values. Our proposed method can be visualized as follows: first, from the ISBSG dataset, we select the attributes and filter it according to the criteria described in the experiments. Each data column corresponding to EI, EO, EQ, ILF, and EIF will be transformed into three values : low, average, and High. Then the data will be removed outliers to reduce noise. The next step is to use the k-fold cross-validation technique to split the data into five folds and run each of these folds against the MLR. Finally, the regression results will be computed to obtain Low, Average, and High values for the function complexity. We named this proposed model is Calibrating Function Complexity in Enhancement Project (CFCEP). **Fig. 3** shows the steps described above in a more intuitive way.

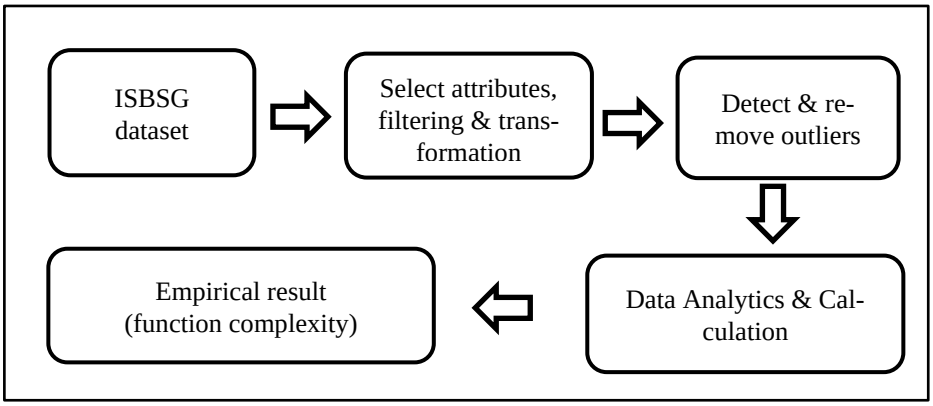

Fig. 3. CFCEP model

For the purpose of this article to find function complexities, we divided the function points values of EI, EO, EQ, ILF, and EIF into 15 values corresponding to low, average, and high for each function point. These 15 values are obtained for each record by approximating the solutions to the 3-variable equation and then taking the mean value. Based on these 15 values, we will calculate the UFP_{CFCEP} value according to the MLR model. Then we calculate the AFP and the Effort values. The MLR model is described as follows:

$$UFP_{CFCEP} = \beta_1 x_1 - \beta 2_2 x_2 - \beta_3 x_3 - \beta_4 x_4 + \beta_5 x_5 + \beta_6 x_6 + \beta_7 x_7 + \beta_8 x_8 - \beta_9 x_9 + \beta_{10} x_{10} + \beta_{11} x_{11} - \beta_{12} x_{12} + \beta_{13} x_{13} + \beta_{14} x_{14} - \beta_{15} x_{15} \quad (12)$$

4 Result and Discussion

In this section, we will review the experimental results. This result estimated through MLR will be compared with using FPA for calculation. Actual results and estimated results will be evaluated based on error criteria MAE, MSE, RMSE.

Table 5 shows us the results of five experimental. As we can see, the result of effort estimation using MLR is much better than the FPA method. With each experiment as well as the average value, the estimated results based on our proposed method are always closer to the actual results than the FPA method. Specifically, the accuracy of this proposed method is 96.13%.

Table 4. Effort estimation results

	SWE	Effort using FPA	Effort using CFCEP
ex-1	3214.661	46599.3	2784.752
ex-2	2991.458	47375.3	2851.248
ex-3	2845.984	48915.68	2942.287

ex-4	3207.625	53258.45	2995.198
ex-5	3168.885	52608.05	3258.606
mean	**3085.723**	**49751.35**	**2966.418**

A smaller value represents a more accurate estimate for the evaluation criteria MAE, MSE, and RMSE [12]. With Table 5, it is clear that the proposed method has much higher accuracy than the FPA method.

Table 5. Evaluation criteria results

	MAE FPA	**MAE MLR**	**MSE FPA**	**MSE MLR**	**RMSE FPA**	**RMSE MLR**
exp_1	43442.04	2500.785	4.42E+09	15246804	66496.63	3904.716
exp_2	44446.85	2017.467	5E+09	10126571	70721.76	3182.227
exp_3	46083.04	2198.764	5.38E+09	11095411	73340.03	3330.978
exp_4	50120.9	2459.952	5.25E+09	15106096	72434.24	3886.656
exp_5	49502.85	2429.726	5.51E+09	12530844	74244.84	3539.893
mean	**46719.14**	**2321.339**	**5.11E+09**	**12821145**	**71447.5**	**3568.894**

The regression model obtained from our experiment is as the following

$$UFP_{CFCEP} = x_1 - 0.58x_2 - 1.41x_3 - 0.49x_4 + 0.26x_5 + 0.31x_6 + 1.02x_7 + 0.34x_8 - 1.20x_9 + 0.08x_{10} + 0.10x_{11} - 0.07x_{12} + 1.45x_{13} + 0.05x_{14} - 1.40x_{15} \quad (12)$$

And the function complexity we also obtained by calculating is shown in **Table 6**

Table 6. Function complexity results

		std	**exp_1**	**exp_2**	**exp_3**	**exp_4**	**exp_5**	**mean**
EI	**L**	3	6.023	7.178	6.628	4.496	7.987	**6.463**
	A	4	-2.445	-1.654	-2.674	-2.859	-1.942	**-2.315**
	H	6	-7.338	-10.939	-8.239	-3.704	-12.093	**-8.463**
EO	**L**	4	-1.159	-1.588	-2.779	-2.856	-1.436	**-1.964**
	A	5	-0.775	2.467	2.507	1.615	0.694	**1.301**
	H	7	3.794	-0.305	1.874	3.298	2.062	**2.145**
EQ	**L**	3	2.468	3.059	2.298	3.927	3.49	**3.048**
	A	4	1.297	0.883	1.3	2.3	1.006	**1.357**
	H	6	-5.825	-6.703	-5.771	-10.336	-7.493	**-7.226**
ILF	**L**	5	0.067	0.433	0.473	0.586	0.474	**0.407**
	A	7	1.078	0.4	0.57	1.625	-0.186	**0.697**
	H	10	-1.049	0.192	-0.363	-2.601	0.208	**-0.722**
EIF	**L**	7	8.107	11.54	11.288	12.103	7.832	**10.174**

	A	10	0.732	0.48	0.374	-0.893	1.628	**0.464**
	H	15	-16.369	-24.64	-22.381	-24.527	-17.3	**-21.043**

The research objective of this study is to propose a new function complexity value for the enhancement project. **Table 6** is our answer to this problem. With this result, we can apply function complexity to enhancement projects with better accuracy.

5 Conclusion

This paper has achieved a new function complexity table for the enhancement project by applying the MLR model to the ISBSG dataset filtered by the suggested criteria. We can see from the experimental results that the average effort achieved with our proposed new method is much closer to the actual recording of SWE compared to the IFPUG FPA method. Specifically, the new method has an accuracy of 96.13%.

Future work will focus on a complete framework for both new development types and enhancement types. In addition, we will be using other methods like artificial neural networks and support vector machines in our model.

Acknowledgment. This work was supported by the Faculty of Applied Informatics, Tomas Bata University in Zlín, under project IGA/CebiaTech/2021/001.

References

1. IFPUG: International Function Point Users Group. http://www.ifpug.org/, access July 2021.
2. A. J. Albrecht, "Measuring application development productivity," Proc. IBM Applications Develop. Symp., pp. 83, 1979.
3. M. Stone, "Cross-validatory choice and assessment of statistical predictions," Journal of the Royal Statistical Society. Series B (Methodological), pp. 111-147, 1974.
4. ISBSG, ISBSG Release 2020 R1
5. Albrecht, A.J., Gaffney, J. E.: Software function, source lines of code, and development effort prediction: a software science validation. IEEE Transactions on Software Engineering, pp. 639–647 (1983).
6. Putnam, L. H: A general empirical solution to the macro software sizing and estimation problem. IEEE Transactions on Software Engineering, pp. 345–361 (1978)
7. Caper. J: Estimating software cost. Mc-Graw-Hill Edition (2007).
8. Boehm, B.W.: Software Engineering Economics.Prentice-Hall, Englewood Cliffs, NJ, USA, (1981).
9. C. Lokan and E. Mendes, "Investigating the use of the chronological split for software effort estimation," IET Softw., vol. 3, no. 5, pp. 422-434, 2009.
10. The Standish Group, https://www.standishgroup.com/ , access July 2021
11. T. Foss, E. Stensrud, B. Kitchenham, I. Myrtveit, "A simulation study of the model evaluation criterion MMRE," IEEE Trans. Softw. Eng., 29 (11) (2003), pp. 985-995
12. B. Kitchenham, S. MacDonell, L. Pickard, M. Shepperd, "What accuracy statistics really measure," IEE Proc. Softw. Eng., 148 (3) (2001), pp. 81-85
13. L. C. Briand, K. E. Emam, D. Surmann, I. Wieczorek and K. D. Maxwell, "An assessment and comparison of common software cost estimation modeling techniques," in International Conference on Software Engineering, 1999, pp. 313-322.
14. V.K. Bardsiri, D.N.A. Jawawi, S.Z.M. Hashim, E. Khatibi, "A flexible method to estimate the software development effort based on the classification of projects and localization of comparisons," Empir. Softw. Eng., 19 (2014), pp. 857-884
15. W. Mendenhall, A second course in statistics: regression analysis. Boston, MA, USA: Pearson Education, Inc., 2012.
16. Moløkken, K., Jørgensen, M.: A review of surveys on software effort estimation. International Symposium on Empirical Software Engineering, 223-231. Retrieved from ACM Digital Library database, (2003).
17. Moløkken - Østvold. K et al.: Project Estimation in the Norwegian Software Industry. A Summary. Simula Research Laboratory (2004).
18. Galorath, D. D., Evans, M. W.: Software sizing, estimation, and risk management: When performance is measured performance improves. Boca Raton, FL: Auerbach (2006)
19. Caper. J: Estimating software cost. Mc-Graw-Hill Edition (2007).
20. Chris F Kemerer : An empirical validation of software cost estimation models, Communications of the ACM, 30(5), 416-429 (1987).
21. Ravichandran, T.: Organizational assimilation of complex technologies: An empirical study of component-based software development. IEEE Trans. Eng. Manag., vol. 52, no. 2, pp. 249–268, (2005).
22. ISO/IEC 20926:2009 (IFPUG), Software and systems engineering -- software measurement -- IFPUG functional size measurement method 2009

23. Marcos, F., Marcelo, F., Sun, V.: Sun, Improvements to the Function Point Analysis Method: A Systematic Literature Review, IEEE Transactions on Engineering Management 62(4):1-12 (2015).
24. Kampstra, P, Verhoef. C: Reliability of function point counts — Department of Computer Science, VU University Amsterdam, Amsterdam, The Netherlands (2010).
25. Kemerer, C. F: Reliability of function points measurement: A field experiment. Commun. ACM, vol. 36, no. 2, pp. 85–97 (1993).
26. Meli, R: Functional metrics: Problems and possible solutions. Proc. 1st Eur. Software Meas. Conf., Antwerp, Belgium (1998).
27. Xia, W., Capretz, L. F., Ho, D.: Neuro-fuzzy approach to calibrate function points, Proc. 8th WSEAS Int. Conf. Fuzzy Syst., pp. 116-119, (2007)
28. Xia, W., Capretz, L. F., Ho, D., Ahmed, F.: A new calibration for function point complexity weights. Inf. Softw. Technol., vol. 50, no. 7–8, pp. 670-683, (2008).
29. Xia, W., Ho, D., Captrez, L. F.: A neuro-fuzzy model for function point calibration. WSEAS Trans. Inf. Sci. Appl., vol. 5, no. 1, pp. 22-30, (2008).
30. Hajri, M. A., Ghani, A. A. A., Sulaiman, M. N., Selamat, M. H.: Modification of standard function point complexity weights system. J. Syst. Softw., vol. 74, no. 2, pp. 195-206, (2005).
31. Ya-Fang, F., Xiao-Dong, L., Ren-Nong, Y., Yi-Lin, D., Yan-Jie, L.: A software size estimation method based on improved FPA. Proc. 2nd World Congr. Softw. Eng., pp. 228-233, (2010).
32. C. Gencel and O. Demirors, "Functional size measurement revisited," ACM Transaction on Software Engineering and methodology, vol.17, no. 3, pp.15.1-15.36, June 2008.
33. R. Meli and L. Santillo, "Function point estimation methods: a comparative overview," in FESMA '99 Conference Proceedings, Amsterdam, 4-8, October 1999.
34. Silhavy, Petr & Silhavy, Radek & Prokopová, Zdenka. (2019). Categorical Variable Segmentation Model for Software Development Effort Estimation. IEEE Access. PP. 1-1. 10.1109/ACCESS.2019.2891878.
35. Prokopova, Zdenka & Silhavy, Petr & Silhavy, Radek, "VAF factor influence on the accuracy of the effort estimation provided by modified function points methods," Annals of DAAAM & Proceedings. 2018, Vol. 29, p0076-0084. 9p
36. J. Wena, S. Lia, Z. Linb, Y. Huc, C. Huang, "Systematic literature review of machine learning-based software development effort estimation models," Information and Software Technology, vol. 54, no. 1, pp. 41-59, 2012.