TECHNIUM

*Technium Vol. 4, No. 9 pp.62-70 (2022)*
*ISSN: 2668-778X*
*www.techniumscience.com*

# Software Size Estimation: A survey

**AESHAA L . AL-SALEEM [1], ASMA'A Y. HAMMO [2]**

[1] Department of Software Engineering College of Computer Science and Mathematics University of Mosul Mosul, IRAQ aisha.20csp7@student.uomosul.edu.iq
[2] Department of Software Engineering College of Computer Science and Mathematics University of Mosul Mosul, IRAQ asmahammo@uomosul.edu.iq

**Abstract**. Software size prediction is a first step to compute information for software project development such as: project cost, effort, schedules, duration and so on. This paper presents the software prediction methods and the efforts of the researchers in this era. Methods such as Source line of Code, Model based estimation, function point and estimation from UML diagrams. For every method, the strength and weakness is presented. Also, it presents the evaluation criteria that is necessary for the researcher to evaluate the accuracy of the methods.This information will help the researcher in the future to choose the most effect method that fit to his/her software project.

**Keywords**. SLOC, Function point, size estimation, UML diagrams, Nesma, COCOMO

## 1. Introduction

Software is an engineering activity, like all engineering activities, measurement is very important in this field. Scientists have classified the metrics of software to two main types: process metrics that deals with characteristics of the processes used for software development, and product metrics also known quality measures that measures size, cost, and complexity [3]. To measure the cost of a program, the size of it should be measured first. Program size is an important parameter for estimating effort and cost. Roger Pressman [1] defines estimation as trying to determine the magnitude of money, effort, resources and time needed to construct any software system. Donatti Guilermo [2] defines the estimation as the art of the approximating the probable cost, extent, quality, quantity, or character of a software based on information available at the time. In my opinion, the two scholars agreed that the definition of estimation should be an art by which the magnitude of cost, effort, and time needed to build any software can be determined based on the available information about that system. Size should be measured not guessed. The poor estimation of the size of the program is one of the main reasons for the failure of the organization, as it is the decisive factor in estimating the effort, the cost, and the schedule. Failure to estimate the size of the program may lead to exceeding the budget or delaying delivery, and this leads to a lack of trust between the customer and the developer. The accuracy of estimating software size depends on: The degree of correctly estimating the volume of the product to be built. The ability to corrected cost estimation, time, and effort based on the estimated size.Stability of product requirements and environment that support software engineering [1]. The main goal of this paper is to show the available researches in the field of software size estimation, present the weakness and strength of each method, and to evaluate the used method and give advices for researchers to choose the most suitable one for their work.

Technium Vol. 4, No. 9 pp.62-70 (2022)
ISSN: 2668-778X
www.techniumscience.com

## 2. Main estimation methods

Many size estimation techniques are available. Figure 1 gives an abbreviation of these techniques. The details are as below:

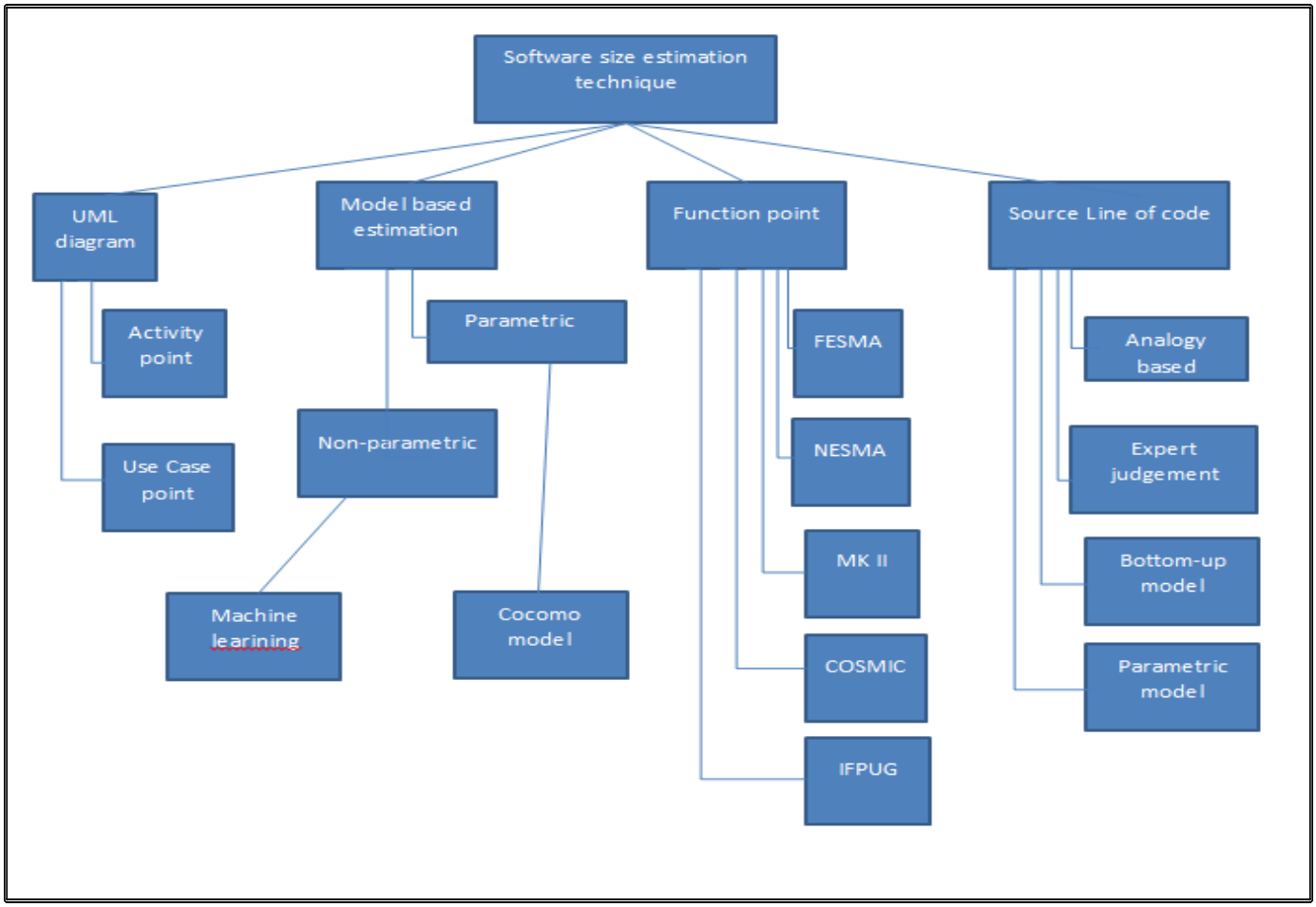


Figure 1 Main Software Size Estimation Techniques

### *2.1. Source line of code SLOC*

Source line of code is the simplest metrics used to get the size of the program. It can be done using:

#### *2.1.1. Analogy based*

In this method the size is predicted by comparing a program with identical programs of known size.

$$SLOC = F * (\text{size of identical program}) \quad (1)$$

F is a factor determined by experience or politics [5].

Technium Vol. 4, No. 9 pp.62-70 (2022)
ISSN: 2668-778X
www.techniumscience.com

### 2.1.2. *Expert Judgment based*

Expert judgment includes making an estimate depends on the experience of one or more experts.

$$SLOC = (X + 4L + P) / 6 \quad (2)$$

X is the smallest possible size.
L is the most likely size.
P is the largest possible size.
Where X, L and P are determined by experts.
So, it is also not of great importance among researchers, as it largely depends on the status and experience of the expert who gives his judgment in the estimation. Hammond [4] claims that expert experience does not lead to a high rate of accuracy in estimating.

### 2.1.3. *Bottom-Up model*

In this model, SLOC is computed from components of known size from a data base. It allows size to be input based on functionality. The user specifies the functions performed by the program and the model obtains a SLOC estimate for each function from the data base.

### 2.1.4. *Parametric Model*

In this model, the input consists of numeral values to calculate software length. Regression analysis and other numerical analysis are used. For example Ikature and Takayangi estimated the size of Cobol program as follows

$$SLOC= -810 +310\ X1 + 1.12X2+553X3 + 5.91X5 + 1.62X7+99.7X8 \quad (3)$$

Where, X1 is “number of input files”, X2 “number of input items”, X3 and X4 are “number of output files and items” respectively.X5 is items of “transaction type reports”, X6 and X7 are number of vertical and horizontal items in two-dimensional table type reports respectively. X8 is no. of calculating processes.
The advantages of SLOC are: simple, widely used, most common size metric, counting can be automated. As for opponents, it depends on the language of Programming, as it is not possible to compare two programs written in different programming languages, and it will be against good software written with shorter code. The project manager is unable to predict the size of the program in the beginning of the Software Development Life Cycle (SDLC) , as it requires the presence of code , and it is not suitable as a measure to evaluate the efficiency of the programmers team [1],[10],[11].

## 2.2. *Function Points (FP)*

It is popular, accurate and useful indirect metrics for measuring software. Compared with (SLOC) the (FP) has proven to be successful in many areas and may be utilized in many stages of the SDLC [11].
Function point models were presented by Albrecht and Gaffny in 1983. Albrecht [12] was the first to invent this method as an alternative to the number of lines in the code in measuring the size of the program from the customer's functional requirements.
The size is estimated in terms of functionality from user’s viewpoint. The traditional model is as in equation 4:

$$BFP = 4\ EI + 5\ EO + 4\ EQ + 10\ ILF + 7\ EIF \quad (4)$$

where EI, EO and EQ are "External Input, Output and Inquiries" respectively. ILF is internal files such as data bases and EIF is the external Interface.

Functional points take into account the number of interfaces, files, and queries from the functional requirements of the customer and are given weights depending on their complexity either (easy, medium or complex). The sum of the weights values is the value of the Unadjusted Function Point (UFP) this value is multiplied by the Complexity Adjusted Factor (CAF) formed from a total of 14 factors, values (0-5) are given according to the importance of each factor, and the result is the value of the adjusted function point (AFP). As define in equations 5 and 6:

$$AFP = UFP * CAF \quad (5)$$

$$CAF = 0.65 + 0.01 * \sum Wi \quad , \ i <= i <= 14 \quad (6)$$

Wi is the weight of 14 factors.

Functional sizing techniques have become the most determined sizing measures, with ISO standards which include NESMA, IFPUG and COSMIC.

NESMA It is one of the methods of estimating the size of the program that depends on the number of functional points in the program and is approved by (ISO 2005) [23]. NESMA acknowledge three function point analysis models: "Detailed function point analysis" DFPA, "Estimated function point analysis" EFPA also called high level function point and "Indicative function point analysis" IFPA. Each of these models is a method in itself. The high level FPA method and the IFPA method do not require detailed user requirements while the results of these two methods are very close to the results extracted from volume estimation by the detailed FPA method. This makes these two methods suitable for application early in the SDLC or if user wants to quickly determine the functional size [24].

The main FPA method supplied the "fundamental basis for the International Function Point Users Group" (IFPUG), ISO standard "International Organization for Standardization, 2009", that is the best generally used FSM [10]. Both NESMA and IFPUG currently use the same philosophy , thought and guidelines with FPA [10][24]. The Mark II way employs the fixed number of data element type (DET) by crossing the restriction of having ultimate border for creation intricacy [10]. The development of the "Common Software Measurement International Consortium" (COSMIC) functional size measurement (FSM) was specifying its goal to extend the functional sizing to real time software. So, in this model each single data movement work in with one of "Entry", "Exit", "Read" or "Write"' and participate equally to total functional size.[10][13]. The use of (FP) can also lead to a significant improvement in the probability of completing projects successfully within the established plan for time and budget.

### *2.3. Model based estimation*

The model-based estimation method is the most prevalent method and is concerned with estimating the size of the program. Its grouped into two types, parametric and non-parametric models. The most known parametric model is COCOMO which was firstly mentioned by Boehm "Boehm's Software Engineering Economics" and then other extensions such as COCOMO II [4], and COSYSMO [5] appeared. Machine learning models are an example of non-parametric models, through which more accurate estimation models can be obtained. The estimation is done by training a network such as a neural network (NN) on historical project data. The accuracy of these models lies in focusing on building the model accurately, while the focus was less on the efficiency of the estimation process [6][7][8][9].

*2.4. Estimation Size Using UML diagrams*

Some researchers used to derive metrics from analysis and design models such as Use case point and activity point[14].

*2.4.1. Activity point*

The activity point model (AP ) is proposed to be a way to predict the size of the software using the activity diagram and modify it using "Technical Complexity Factors" (TCF), "Environment Complexity Factors" (ECF), and "People Risk Factors" (PRF). Unadjusted size is calculated from "Total Activity Point" (TAP), and "Total Activity Complexity Weight" (TACW). TAP is a factor, which gives size at low level, including TAW and TPW. The level of TAW and TPW is grouped into 3 levels :( simple, average, and complex). TAP is calculated as in equation 7. A number of transactions in use case diagram is substituted by a number of activities or incidents. A number of scenarios are indicated by a number of paths in activity diagram. TAW is computed as in equation 8. In the same way, TPW is calculated as in equation 9. TACW is an operator that explains a size at high level of activity diagram. The overall complexity explains size as Activity Complexity (AC) and calculated as in equations 10,11 and 12 [17].

$$TAP = TAW + TPW \quad (7)$$

$$TAW = \sum \text{NA level} * \text{weight} \quad (8)$$

$$TPW = \sum \text{NP level} * \text{weight} \quad (9)$$

$$TACW = \sum \text{AC level} * \text{weight} \quad (10)$$

$$UAP = \sum_{i=1}^{n} TAPi + TACW \quad (11)$$

$$AP = UAP * TCF * ECF * PRF \quad (12)$$

*2.4.2. Use case point*

This method was adopted by researchers to estimate the size of the software, and this method depends on the number of actors and the number of use cases and to calculate the number of use cases requires four variables are required: "Unadjusted Actor Weights" (UAW), "Unadjusted Use Case Weights" (UUCW), "Technical Complexity Factor" (TCF), and "Environmental Complexity Factors" (ECF) ,are necessary to calculate UCP. Weights of actors and use cases are based on three complexity groups i.e.simple, average, and complex. The use case description is applied to calculate the "Success Transactions" (ST) and "Alternate Transactions" (AT) of a use case. The "Total Transactions" (Ti) of a use case i are the total of STi and ATi. UUCW is the total of all weighted use cases. Even though UAW, TCF, and ECF contribute to software size, it participate more to the size of the system as in equations 13 till 16 [18].

$$ST = \sum_{i=1}^{n} s - ti \quad (13)$$

$$AT = \sum_{i=1}^{n} a - ti \quad (14)$$

$$Trans = \sum_{i=1}^{n} s - ti + a - ti \quad (15)$$

$$UUCW = \sum_{i=1}^{3} ni * wi \quad (16)$$

## 3. Evaluation criteria

Usually, "Mean Magnitude of Relative Error" (MMRE) is used to scale the precision of the size prediction methods [19], "Percentage relative error deviation" Pred(x) [20] and "Mean Absolute

Residual" (MAR) [21]. Both MMRE and Pred(x) are computed using a metric called "Magnitude of Relative Error" (MRE). Equation (17) shows the formula used to calculate MRE of a data point I whereas yi denotes the actual size and yi^ denotes the estimated size. MMRE of a dataset comprising of n data points is taken out by calculating the mean of all n MREs as shown in equation (18). Pred(x) takes the proportion of predicted values that are within x% of the actual values [22] and is computed using equations 19 and 20

$$MREi = \frac{|yi - \widehat{yi}|}{yi} \quad (17)$$

$$MMRE = \frac{1}{n}\sum_{i=1}^{n} MREi \quad (18)$$

$$Pred\ (x) = \frac{1}{n}\sum_{i=1}^{n} \begin{cases} 1 & MREi \leq \frac{x}{100} \\ 0 & other\ wise \end{cases} \quad (19)$$

$$MAR = \frac{1}{n}\sum_{i=1}^{n} |yi - \hat{y}| \quad (20)$$

## 4. Related works

Table 1 presents the most recently researches for predicting the size of a program. It explains the used method for predicting and the dataset (if available.

**Table 1.** Survey of Software Size estimation method

| Id | Research | year | Method | Data set that used |
|---|---|---|---|---|
| 1 | "A case study in COSMIC functional size measurement: Angry Bird Mobile Application in Open Systems (ICOS)" [25] | 2013 | Use the UML representation in the COSMIC FSM calculation. | case study in mobile game |
| 2 | "A measurement method for sizing the structure of UML sequence diagrams Information and Software Technology" [16] | 2015 | Use a set of procedures to scale sequence diagrams (UML)with high accuracy using COSMIC rules | The software of the real time system "Rice Cooker" |
| 3 | "ESSE: An early software size estimation method based on auto-extracted requirements Features" [26] | 2016 | Suggesting a new route for predicting software sizing early in the SDLC in which the projects scaling model is trained by a regression algorithm. | real industrial |

| | | | | |
|---|---|---|---|---|
| 4 | "Effort estimation for agile software development: Comparative case studies using COSMIC functional size measurement and story points" [27]. | 2017 | Adopting the story point and COSMIC In a study that compared the effectiveness of estimating efforts based on Cosmic and the story point, the results showed the efforts based on Cosmic more accurate, so it is considered an appropriate measure for organizations that adopt agile software | N\A |
| 5 | "An effective approach for software project effort and duration estimation with machine learning algorithms" [28]. | 2018 | Machine learning algorithms are used (Support Vector Machines, Neural Networks and Generalized Linear Models) to spread and maintain machine learning and reduce the gap between recent research results and applications within organizations. | ISBSG |
| 6 | "Differential evolution using homeostasis adaption based mutation operator and its application for software cost estimation" [29] | 2021 | Use (COCOMO) for optimizing the tuning parameters. And Using the different artificial intelligence (AI) approaches searched | N/A |
| 7 | "Using COSMIC method for web app effort estimation" [30] | 2018 | Using COSMIC full function point | 19 Webapps with their conceptual models |
| 8 | "A deep learning model for estimating story points" [31] | 2019 | Suggesting a method for predicting story points based on " long short-term memory and recurrent highway network". | 16 open source projects |
| 9 | "Efficiency Improvement of Function Point-Based Software Size Estimation With Deep Learning Model" [11] | 2020 | A BiLSTM-CRF model was applied for function point recognition, that integrate RNN-based BiLSTM "bidirectional long short-term memory" neural network and CRF (conditional random field) model to treat various estimation methods. | 29 real projects |
| 10 | "Using Actors and Use Cases for Software Size Estimation" [14]. | 2021 | This method depends on the number of actors and use cases for estimate the size of software | Dataset was taken from the web site "(https://data.mendeley .com/datasets/ 2rfkjhx3cn/1)" |
| 11 | "Improving the Accuracy of Early Software Size Estimation Using Analysis to-Design Adjustment Factors (ADAFs)" [18]. | 2021 | Estimate the size from the analysis phase and the design phase and compare between them | 29 C++ industrial subsystems<br>8 C++ industrial systems<br>12 Java student projects |

## 5. Results and conclusion

The studies and works that have been made on the Software Size Estimation is presented in this research. Mainly, that there is no method that gives the accurate size estimation. SLOC is an easy and suitable for calculating the software size after writing the software but its not suitable for early stages in Software Life Cycle. To estimate the size of the program in the early stages of the software life cycle, FP or UML diagrams can be applied, such as use case diagram, activity diagram, class diagram. By applying the above methods, the real size of the software will be produced because it will represent the number of functions performed by the system.
Finally, we encourage the researchers and practitioners to use a combination of multiple approaches instead of one to give an accurate result.

## References

[1] Roger S Pressman and Bruce R Maxim,2020 “software engineering a practitioners approach”
[2] Donatti, Guliermo S., (2005) "Software Development Effort Estimation Through Neural Networks" , Faculty of Mathmatics, Astronomy and physics, codoba national university May 2005
[3] Singh, G., Singh, D. and Singh, V., 2011. A study of software metrics. IJCEM International Journal of Computational Engineering & Management, 11(2011), pp.22-27.
[4] B. Boehm, C. Abts, A. W. Brown, S. Chulani, B. K. Clark, E. Horowitz, R. Madachy, D. J. Reifer, and B. Steece, Cost Estimation With Cocomo II. Upper Saddle River, NJ, USA: Prentice-Hall, 2000
[5] R . Valerdi, B. W. Boehm, and D. J. Reifer, ``COSYSMO: A constructive systems engineering cost model coming of age," in Proc. INCOSE Int. Symp., vol. 13, no. 1. Hoboken, NJ, USA: Wiley, 2003, pp. 70_82.
[6] Y. Singh, P. K. Bhatia, and O. Sangwan, ``ANN model for predicting software function point metric," ACMSIGSOFT Softw. Eng. Notes,vol. 34, no. 1, pp. 1_4, Jan. 2009.
[7] F. de Barcelos Tronto, J. D. S. da Silva, and N. Sant'Anna, ``An investigation of arti_cial neural networks based prediction systems in software project management," J. Syst. Softw., vol. 81, no. 3, pp. 356_367, Mar. 2008
[8] K. V. Kumar, V. Ravi, M. Carr, and N. R. Kiran, ``Software development cost estimation using wavelet neural networks," J. Syst. Softw., vol. 81, no. 11, pp. 1853_1867, Nov. 2008.
[9] P. Pospieszny, B. Czarnacka-Chrobot, and A. Kobylinski, ``An effective approach for software project effort and duration estimation with machine learning algorithms," J. Syst. Softw., vol. 137, pp. 184_196,Mar.2018,doi:10.1016/j.jss.2017.11.066.
[10] Morrow, P., 2018. Software sizing for cost/schedule estimation (Doctoral dissertation, Ulster University).
[11] KUI ZHANG , XU WANG , (Member, IEEE), JIAN REN 1, AND CHAO LIU, Efficiency Improvement of Function Point-Based Software Size Estimation With Deep Learning Model ,2020
[12] Albrecht, A., J., (1985), Function point help managers assess application, maintenance value, Computerworld Special Report on Software productivity, CW communication, pp. SR20-SR21
[13] S. Bagriyanik and A. Karahoca, ``Automated COSMIC function point measurement using a requirements engineering ontology," Inf. Softw. Technol., vol. 72, pp. 189_203, Apr. 2016.
[14] R. Silhavy, P. Silhavy, and Z. Prokopova, ``Using actors and use cases for software size estimation," Electronics, vol. 10, no. 5, pp. 1_20, 2021.
[15] R. Silhavy, P. Silhavy, and Z. Prokopova, ``Analysis and selection of a regression model for the use case points method using a stepwise approach," J. Syst. Softw., vol. 125, pp. 1_14, Mar. 2017.

[16] Sellami A, Hakim H, Abran A, and Ben-Abdallah H 2015 A measurement method for sizing the structure of UML sequence diagrams Information and Software Technology vol. 59 pp. 222-32.
[17] S Densumite and P Muenchaisri "software size estimation using activity point "2017
[18] MARRIAM DAUD AND ALI AFZAL MA , " Improving the Accuracy of Early Software Size Estimation Using Analysis-to-Design AdjustmentFactors (ADAFs)"
[19] T. Foss, E. Stensrud, B. Kitchenham, and I. Myrtveit, ``A simulation study of the model evaluation criterion MMRE," IEEE Trans. Softw. Eng., vol. 29, no. 11, pp. 985_995, Nov. 2003.
[20] D. Port and M. Korte, ``Comparative studies of the model evaluation criterions MMRE and pred in software cost estimation research," in Proc. 2nd ACM-IEEE Int. Symp. Empirical Softw. Eng. Meas. (ESEM), Oct. 2008, pp. 51_60.
[21] M. Shepperd and S. MacDonell, ``Evaluating prediction systems in software project estimation," Inf. Softw. Technol., vol. 54, no. 8, pp. 820_827, Aug. 2012

[22] L. C. Briand, J. Wüst, J. W. Daly, and D. V. Porter, ``Exploring the relationships between design measures and software quality in objectoriented systems," J. Syst. Softw., vol. 51, no. 3, pp. 245_273, May 2000
[23] Morrow, Philip, F. George Wilkie, and I. R. McChesney. "Function point analysis using NESMA: simplifying the sizing without simplifying the size." Software Quality Journal 22.4 (2014): 611-660
[24] https://nesma.org/wp-content/uploads/2018/05/Nesma-on-sizing-1-FPA-1.pdf
[25] Abdullah, Nur Atiqah Sia, Nur Ida Aniza Rusli, and Mohd Faisal Ibrahim. "A case study in COSMIC functional size measurement: angry bird mobile application." In 2013 IEEE Conference on Open Systems (ICOS), pp. 139-144. IEEE, 2013.
[26] Zhang, Cheng, Shensi Tong, Wenkai Mo, Yang Zhou, Yong Xia, and Beijun Shen. "Esse: an early software size estimation method based on auto-extracted requirements features." In Proceedings of the 8th Asia-Pacific Symposium on Internetware, pp. 112-115. 2016.
[27] Salmanoglu, Murat, Tuna Hacaloglu, and Onur Demirors. "Effort estimation for agile software development: Comparative case studies using COSMIC functional size measurement and story points." In Proceedings of the 27th International Workshop on Software Measurement and 12th International Conference on Software Process and Product Measurement, pp. 41-49. 2017.
[28] Pospieszny, Przemyslaw, Beata Czarnacka-Chrobot, and Andrzej Kobylinski. "An effective approach for software project effort and duration estimation with machine learning algorithms." Journal of Systems and Software 137 (2018): 184-196.
[29] Singh, Shailendra Pratap, Vibhav Prakash Singh, and Ashok Kumar Mehta. "Differential evolution using homeostasis adaption based mutation operator and its application for software cost estimation." Journal of King Saud University-Computer and Information Sciences 33, no. 6 (2021): 740-752
[30] Ahmed , A.T. and Taha, D.B., Webapp Effort Estimation using Cosmic Method. International Journal of Computer Applications,(2018): 975, p.8887.
[31] Choetkiertikul, Morakot, Hoa Khanh Dam, Truyen Tran, Trang Pham, Aditya Ghose, and Tim Menzies. "A deep learning model for estimating story points." IEEE Transactions on Software Engineering 45, no. 7 (2018): 637-656.