Received: May 6, 2023. Revised: June 15, 2023.

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Implementation of COSMIC Function Points (CFP) as Primary Input to COCOMO II: Study of Conversion to Line of Code Using Regression and Support Vector Regression Models

Sholiq Sholiq[1,2] **Riyanarto Sarno[1]*** **Endang Siti Astuti[3]** **Muhammad Ainul Yaqin[4]**

[1]Department of Informatics, Institut Teknologi Sepuluh Nopember Surabaya, Indonesia
[2]Department of Information Systems, Institut Teknologi Sepuluh Nopember Surabaya, Indonesia
[3]Department of Business Administration, Brawijaya University Malang, Indonesia
[4]Department of Informatics, Universitas Islam Negeri Maulana Malik Ibrahim Malang, Indonesia
* Corresponding author's email: riyanarto@if.its.ac.id

Abstract: In COCOMO II, the primary input for estimating development effort in person month (PM), duration, and cost is the size of the software. Until now, there are two ways to get the size, namely (1) size is estimated using a line of code of software, and (2) size is estimated using unadjusted function points (UFP), which is one of the functional size measurements (FSM). In this study, we added a new way to obtain the size as the primary input in COCOMO II, namely with COSMIC function points (CFP). CFP has several advantages compared to other FSMs, including UFP. Therefore, like UFP, CFP is converted first to LOC, so the conversion equation must be obtained first. We applied four models to get the conversion functions: Ordinary least squares regression (OLSR), support vector regression (SVR) with linear, polynomial, and Gaussian kernel functions. The four models were applied using a dataset from small-scale business application software in Java. The results showed that PM estimation using the CFP model as the primary input produced better accuracy based on MMRE and Pred (0.25), namely 17%-19% and 67%-80%, than the UFP model on the COCOMO II of 135% and 10%.

Keywords: COCOMO II, Software size, CFP, FSM, LOC.

1. Introduction

Measuring software cost is essential for several purposes. The software project manager can control the project from design to implementation and evaluation. Four primary performances must be measured and controlled by managers in order to control the project, namely [1]: (i) project delivery for time and budget, (ii) project productivity, (iii) project speed, and (iv) product size and quality. Software cost is measured in the software development process's initial phases for project planning.

One of the most popular methods used to measure software costs is the constructive cost model (COCOMO), a software cost estimation model introduced by Barry Boehm in 1981 [2]. It is a model that uses a set of equations to estimate the cost, effort, and time required to develop a software system based on a set of input parameters. COCOMO is a well-known model widely used in the software industry for many years. It is a simple model that uses a single equation to estimate the cost of software development based on the project size. The size of the project is measured in lines of code or function points, which measure the functionality provided by the software [2].

COCOMO II (constructive cost model II) is an updated version of the original COCOMO model. COCOMO II was designed to be a more comprehensive and accurate cost estimation model for software development projects. Now, COCOMO II is still in use until the latest COCOMO release comes out; until now, COCOMO III is still in the development process and has not yet been released [3]. The person-months (PM) formula in COCOMO II is used to estimate the effort required to complete

Received: May 6, 2023. Revised: June 15, 2023.

a software development project. The formula is expressed as [2]:

$$PM = A \cdot size^{E} \cdot \prod_{i=1}^{n} EM_i \quad (1)$$

Where:

- PM is the estimated effort in person months required to complete the project.
- A is a constant that depends on the specific project and the organization carrying out the development. It is typically determined based on historical data from previous projects. A constant default equal to 2.94.
- Size is the estimated size of the software project in lines of code or another appropriate size measure.
- E is an exponent that reflects the degree to which effort increases as size increases. It is typically determined based on historical data and ranges from 0.91 to 1.2 for different types of projects.
- EM_i are effort multipliers that reflect the impact of various factors on the effort required for the project. There is a total of n such factors, and they can be classified into five categories: product, personnel, project, platform, and process.

We highlight how the variable size is obtained in COCOMO II. The size variable is a crucial input to the COCOMO II formula and estimates the effort required to complete the project. To determine the size of the project, the COCOMO II model uses one of two methods:

1. Lines of code (LOC): In this method, the size of the software is estimated based on the number of lines of code that will be written. This method assumes that the size of the software project is directly proportional to the amount of code that needs to be written.
2. Unadjusted function points (UFP): In this method, the size of the project is estimated based on the functionality that the software system will provide. UFPs are a measure of the functionality provided by the system and are calculated based on the number of inputs, outputs, inquiries, files, and interfaces the system will have. If using UFP, it must be converted into a LOC before being used as a COCOMO II input. So far, Boehm [7] has used UFP to get the size, and then the size in the UFP is converted using a conversion ratio that depends on the programming language used during software development. If using Java, the conversion rate is 53 LOC per UFP.

In the early phases of a software development project, especially for new software development, using input LOC is difficult because LOC is only obtained after the software project is completed. Therefore, it is more realistic to use the UFP input. UFP is one of the techniques to get the size of software based on its functionality. Today, the measurement methods that are widely used are those based on functionality, so they are called functional size measurements (FSM) [4, 5]. FSM is based on functional user requirements (FUR) are generally used to estimate efforts in the software project phase's initial stages.

Some other FSM techniques derived from function points include COSMIC function points (CFP), use case points (UCP), NESMA, FISMA, MK II, and others [6, 7]. The following references give some advantages of using CFP compared to other FSM methods. The functional sizing method using FSM is a huge success compared to other methods as it deals with different types of software, including business applications and mobile applications etc. [8]. Authors in Ref. [9] tested the accuracy and reproducibility between function points analysis (FPA) and CFP. The results showed that the performance did not differ significantly in accuracy and reproducibility. However, in any case, the procedure in CFP is more straightforward than in FPA. Also, CFP is better suited for a broader application domain than others of FSM [10].

Therefore, we propose using CFP for primary input as the size for COCOMO II. It is to expand the range of COCOMO II to receive primary size inputs other than LOC and UFP, as described above. Almost the same as using UFP as the size for the primary input to COCOMO II. CFPs must also be converted first to LOC. Several ways have been developed to obtain estimated LOC in the early phases of software development by converting FSM to LOC using conversion ratios, known as backfiring. The study investigated the relationship between CFP and LOC using a dataset of fourteen projects constructed with C++ programming. Another study has found an association between CFP and LOC in Java programming for mobile applications carried out in the studies [11, 12]. Lind and Heldal [13] tested the relationship between CFP and LOC in C++ programming using datasets containing fifteen components for the automotive industry. The study indicated that the correlation between CFP and LOC was relatively moderate. There are some limitations in previous research. First, the correlation between CFP and LOC shows varying values, some are strong, and some are weak. Second, the dataset used still has few objects, and third, the technique used still uses simple linear regression. Studies have yet to convert CFP to LOC for business application software. In this study, we conducted an experimental study to convert CFP to LOC in Java programming. Therefore, to make this research more focused, we intend to answer the following questions: (1) How to convert from CFP to LOC in Java programming for business application software? (2) Is CFP, as the primary input of COCOMO II, better than UFP?

The correlation between this study and the Journal lies in the alignment of their subject matter, methodology, and objectives. (1) This subject matter aligns with the interests of the international journal of intelligent engineering and systems, which actively seeks research related to intelligent techniques and methodologies applied to various engineering domains. By exploring the integration of CFP into COCOMO II and investigating the conversion to lines of code using regression and SVR models, this study addresses a topic relevant to the Journal's scope. (2) This methodology in this study compares machine learning algorithms and statistical techniques, which aligns with the Journal's focus on intelligent engineering systems. The use of these models demonstrates the application of intelligent techniques in software cost estimation, which is of interest to the readership of the Journal. (3) The paper's objective is to evaluate the effectiveness of using CFP as the primary input to COCOMO II and to compare the performance of regression and SVR models in estimating effort. This objective aligns with the Journal's interest in research that provides insights into intelligent engineering systems. By presenting empirical results and discussions on the performance of the models, the paper contributes to the understanding of software cost estimation techniques, which is valuable for academia and industry.

In the remaining part of this paper, we arrange as follows. Section two describes the related work, briefly explaining previous studies related to this study. Section three includes our methods for converting CFP to LOC in Java programming and collecting data. Section four describes the results obtained and their discussion about them. Moreover, section five contains the conclusion, the limitations, and the recommendations for the subsequent studies.

2. Related work

Authors of the refs. [14–17] developed a tool to automate COSMIC FSM for Java applications using mapping rules techniques. The study in [14] requires source code input for business applications based on Java programming; the tool created is named cosmic solver. Likewise, the study in [15] requires source code input from Java business applications with a three-tier architecture. The study has deviations with manual testing of 94%. The tool created for the study in [16] also requires source code input from Java applications that use the Spring MVC framework. In contrast, the tool in the study [17] requires input in the form of UML artefacts. Unfortunately, the previous studies above required input in the form of source code. The source code is obtained when the project has been coded.

Kazi and Kazi [18] used CFP to estimate project duration using data flow diagram (DFD) input. In general, the techniques used are CFP and DFD mapping. Meanwhile, the study's authors [19] created a rule mapping from CFP to a language intended to automate programming language compilers. CFPs are also considered suitable for broader application domains, such as web and mobile applications [20]. CFPs can also reportedly be developed for effort estimation for mobile applications in agile environments [21].

The previous studies related to CFP into LOC conversion are given in Table 1. The authors in the paper do the conversion from function points to LOC [22], and which conversion ratio is usually called Backfiring. The studies used neuro-fuzzy to get the conversion ratio from function points to LOC using the dataset from ISBSG release 9. Meanwhile, authors of Ref. [23] investigated the relationship between FSM (in IFPUG and CFP) and code- size (in Kbyte and LOC) using a dataset from ISBSG 2007 Repository release 10 with 14 projects selected (using the C++ programming language). Linear regression analysis obtained a LOC/CFP conversion ratio of 6.03 with a low-medium correlation (0.417). CFP and line of code (LOC) investigations in Java Programming for mobile-based applications were carried out in the studies [11, 12]. The studies reported that CFP strongly correlates (0.89) with other actions, namely Kbyte and LOC. Another study by Lind and Heldal [13] also tested the relationship between FSM expressed in CFP and the code size stated in Kbyte and LOC using a dataset containing 15 software components in the automotive industry formed using the C++ language. A simple linear regression analysis found that the correlation between CFP and Kbyte is robust, while the correlation between CFP and LOC is low-moderate. The author of Ref. [24] tested the effectiveness of using FPA and CFP for web-based projects. High-level FPA and CFP are effective in estimating project effort.

There are some limitations in previous studies regarding CFP to LOC conversion. First, the correlation between CFP and LOC shows varying

Table 1. Summary of previous studies like this study

Paper	Method used	Dataset/ output	Limitations
[22]	Hybrid Neuro-Fuzzy	ISBSG release 9/ Conversion ratio	The LOC/function points conversion ratio is obtained but not specific to a particular programming language.
[23]	Linear regression	ISBSG release 10 & own data set/ LOC/CFP or LOC/IFPUG-FP ratio.	• Low correlation & too wide range of LOC/CFP and LOC/IFPUG-FP • There are too few objects in the own dataset
[11] [12]	Nonparametric statistics Spearman's rho	13 Android mobile applications/ Spearman Rho correlation	• Not showing the value or conversion equation from CFP to LOC. • The dataset has too few objects
[13]	Linear regression	The dataset in GM contains 15 components of software	• The correlation between CFP and LOC is low, • The dataset has too few objects.
[24]	Simple linear regression	25 Web applications from an Italian medium-sized software company/ High-Level FPA are effective & all three CFP approaches are effective	• It is used as a web application, so it needs to be tested for its effectiveness on other platforms. • Not specific to just one programming language
[3]	Not mentioned	ISBSG/ 1 CFP is equal to 5.53 LOC	• Not discussed how to convert CFP to LOC. • No specific programming language.
This study	Linear regression and SVR	Own dataset/ conversion equation	-

Abbreviations: ISBSG = international software benchmarking standards group, FPA = function points analysis, LOC= source line of code, CFP= COSMIC function points.

Table 2. The short project profile is used in this study

ID	Project name	Sort description	Technology used	∑ Function Process
A	Cooperative application	Java-based application developed was for employee cooperatives. The application comprises an electronic unit module, a SIPA module, and the main module. The cooperative had 6,000 members.	Netbeans, Java, My SQL, Astah UML	32
B	Store application	The store application consisted of a point of sales, warehouse, purchasing, and primary modules.	Netbeans, Java, My SQL, Astah UML	21
C	Mini hospital	Java-based application developed was for a health clinic such as a mini-hospital. The application consists of outpatient and inpatient modules.	Eclipse, Java, MySQL	36
D	Medical store application	The Java-based desktop application is for a medical shop.	Eclipse, Java, MySQL	25

values, some are strong, and some are weak. Second, the dataset used still has few objects, and third, the technique used still uses simple linear regression. Therefore, this study is intended to cover the weaknesses of previous studies to determine the conversion from CFP to LOC. In this study, besides linear regression, Support Vector Regression (SVR) is also used, which is a technique adapted from machine learning for classification problems. This SVR is the application of the SVM algorithm in the regression case.

3. Method

The method consists of three sections; the first contains the dataset gathering, which contains definitions of operational variables, data sources and methods for obtaining datasets, and CFP calculations and explanations. The second section contains the conversion method from CFP into LOC, and the third section compares with UFP Input into COCOMO II. All three sections are given in succession below.

3.1 Dataset gathering

Data gathering consists of defining research variables, data source and gathering methodology, CFP calculation, and dataset obtained. Each of them is discussed in order.

3.1.1. Definition of research variables

Operational definitions of variables are used to facilitate researchers in collecting data. In this study, three variables are used. First is CFP as the input variable, and second is UFP as the input variable. The third is LOC as the output variable. For the need to build a conversion model, we use CFP and LOC variables, while the UFP, CFP, and LOC are used to compare the accuracy between inputs using CFP and UFP on COCOMO II.

3.1.2. Data source and gathering methodology

Quantitatively, the software project dataset is usually limited compared to other datasets. To get a dataset containing software applications with CFP, UFP, and LOC attributes, we need a source of software documentation in the form of functional user requirements (FUR) to calculate the CFP and UFP of the software and the source code to get the LOC.

The dataset consists of FP objects of the four projects, as shown in Table 2. The four applications

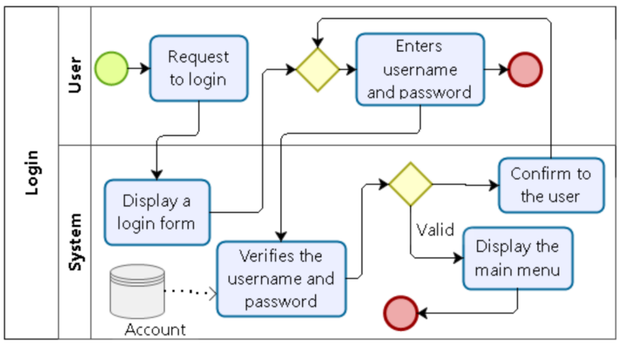


Figure 1. The BPMN diagram for describing user login

are small-scale businesses created by the software industry in Indonesia. Each application is developed using Netbeans and Eclipse IDE environments, JDK, and the MySQL database.

Three variables are observed from the dataset objects: CFP and UFP as the independent variable and LOC as the dependent variable. To get the attributes for an object of the dataset, we do the following steps:

- We identify the function processes and list them on a table. Then we generate detailed BPMN-level diagrams for each function process using the Bizagi modeller to derive CFP and UFP attributes based on the application's function process. CFP and UFP can be calculated based on the BPMN diagram that has been made. Examples of identifying function processes, compiling detailed BPMN level diagrams, and calculating CFP and UFP are in section 3.3.1.
- We grouped application software source code according to their function process. Then, we calculate the LOC for each functional process using the free download LocMetrics for Windows to obtain the LOC attribute [25].

3.1.3. CFP calculation and dataset obtained

As an example of CFP and UFP calculation, we take one of the function process that almost all applications have, namely login. Login is performed by the user communicating with the application system. The details of communication between the user and the system are given in Fig. 1. The process starts with the user requesting to enter the system; the login form is displayed. The following process is for the user to enter the username and password, and then the system will verify the account. If the username and password are correct, the process continues by displaying the main menu. If the username and password are incorrect, the system will notify that the username or password is incorrect, and then the user enters the username and password again.

Based on the BPMN diagram for CFP calculation, we can obtain the amount of input data (E), output data (X), data read from files/tables (R), and data written to files/tables (W). As an example of calculating CFP, referring to the BPMN diagram from Function Process login in Fig. 1, we get E=2, X=1, R=1, and W=0 so that the CFP value can be obtained as CFP=2+1+1+0= 4. Meanwhile, UFP calculations are also based on BPMN Diagrams. UFP calculation rules follow this Ref [2]. For example, in Fig. 1, we get a simple type of external input 1 time, a simple type of internal logical file (ILF) 1 time, and a simple type of external inquiry (EQ) 1 time. At the same time, external output (EO) and external interface files (EIF) are missing. For simple type, EI has a weight of 3, simple type ILF has a weight of 7 and simple type EQ weights 3, so for Login, FP has UFP=3+7+3+0+0=13.

The total function processes for the four software projects is 114, with details of project ID=A having 32 function processes, project ID=B having 21 function processes, project ID=C having 36 function processes, and project ID=D having 25 function processes. So, the total number of function processes for the four projects is 114. Therefore, the dataset obtained has 114 instances. Furthermore, after obtaining the LOC and CFP data, including E, X, R, and W, we put these datasets in this link: https://intip.in/k1HG. In contrast, UFP is used later to compare the accuracy of effort calculations in COCOMO II using CFP and UFP quantities as the primary input.

3.2 Proposed conversion method

This section consists of modelling data using linear regression, modelling data using SVR, and creating conversion function. The discussion of these subsections is given below.

3.2.1. Modelling data using linear regression

In this previous study [20] also used more analysis with linear regression either simple linear regression or multiple linear regression to build an effort prediction model with CFP. Ordinary least square regression (OLSR) is one of the methods in regression analysis to determine the effect of independent variables on independent variables. Analysis with OLSR requires fulfilling a classical assumption known as the best linear unlimited estimator (BLUE). BLUE stated that the data must be a normal distribution, homoscedasticity, no multicollinearity, and no autocorrelation. The OLSR method will meet the conditions of BLUE if it meets all these assumptions. However, if one or more premises are not fulfilled, the estimation results obtained cannot fulfil the BLUE condition.

For this reason, the test of these classic assumptions includes [26]: (1) The normality test aims to know that the data are typically distributed and independent. (2) The multicollinearity test is a condition with a perfect linear relationship or near perfect between the independent variables in the regression model. (3) The autocorrelation test is the relationship between the residuals of one observation and another. (4) A heteroscedasticity test is a condition where the error term does not have a constant variant for all observations. After the BLUE condition is obtained, the OLSR implementation is carried out on the dataset using the academic version of the MATLAB R2020a application.

3.2.2. Modelling data using support vector regression

Support vector regression (SVR) is an SVM development for regression cases. SVR has proven to be an effective tool in many applications [27] and produced models that have high accuracy [28]. In the case of regression, the output used is a real or continuous number. Authors in [29] illustrated SVR with a training data set {(x_1, y_1), ..., (x_n, y_n)} X × R where X defines the input space pattern (let us say X = Rd). In SVR, the goal is to find a function f(x) that has the largest deviation from the actual target y_i for the entire training data and, simultaneously, look for a uniform function as possible. Thus, all errors (the difference between the function output and the actual target) will be ignored if the value is less than ε, but will not accept all errors greater than ε. The formula for SVR is given in (2) [29].

$$f(x) = \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)g(x_i - x) + b \tag{2}$$

Where,
α_i, α_i^*: Lagrange multiplier,
$g(x_i - x)$: Kernel function, and
b: Constant

SVR uses kernel functions to transform non-linear input into a feature space of higher dimensions which is then solved linearly. We must choose the kernel function to use in the SVR model. The kernel functions that are often used in the SVR method are given in (3) to (5) [29]:

$$\text{Linear kernel } g(x_i - x) = x_i . x_j \tag{3}$$

$$\text{Polynomial kernel } g(x_i - x) = ((x_i . x_j) + 1)^d \tag{4}$$

$$\text{Gaussian kernel } g(x_i - x) = exp\ (-|x_i - x_j|)^2 \tag{5}$$

The kernel functions used in this study are Linear, Polynomial, and Gaussian. The modelling is carried out using the help of the academic version of the MATLAB 2020a application. After this modelling is completed, the following process, such as creating hyperplane functions, plotting curves, and predicting LOC, can be done. We will explain these in the next section.

3.2.3. Creating conversion function

After applying the OLSR and SVR models (with three kernel functions) using the dataset, we generate a conversion equation that maps CFP to LOC for the four models already obtained. The conversion function is a linear equation in the OLSR model, and the hyperplane function is in the SVR model. With these four equations, the LOC can be predicted.

3.3 Comparison with UFP input on COCOMO II

In software estimation, the mean of magnitude relative error (MMRE) [30, 31], and Pred (x) [31] be used for the evaluation of the most likely estimate of effort. Meanwhile, magnitude relative error (MRE) is obtained using Eq. (6). The average of all MRE observations becomes MMRE, for which the MMRE formula is shown in Eq. (7), while Pred (x) uses Eq. (8). Actual_Effort is calculated using formula (1) with the actual LOC, while Estimated_Effort is calculated using Eq. (1) with the LOC obtained from the conversion function. Therefore, four Estimated_Effort out of four functions have been obtained. Of the four models, we compare the most accurate according to MMRE and Pred.

$$MRE_i = \frac{|Actual\_Effort_i - Estimated\_Effort_i|}{Actual\_Effort_i} \quad (6)$$

$$MMRE = \frac{1}{N}\sum_1^N MRE_i \quad (7)$$

$$Pred(x) = \frac{1}{N}\sum_{i=0}^{n}\begin{cases}1, & if\ MRE_i \le x\\ 0, & otherwise\end{cases} \quad (8)$$

4. Results and discussion

The results and discussion sections are divided into several subsections, namely a brief description of the dataset, implementation of Linear Regression, implementation of SVR, a conversion function, comparison between UFP and CFP as main on COCOMO II, and ended with a discussion about the results.

4.1 Brief description of dataset

The purpose of presenting descriptive data in this section is to provide a concise description and brief information from the data used in this study. This section contains data distribution on each input variable, CFP, and output variable, LOC. This study uses 114 instances of function process in four projects. Brief information about the CFP and LOC variables is given in Table 3, including the number of observations (instances), min, max, mean, median, mode, standard deviation, and range.

Table 3. Data description

Measure	CFP	LOC
Observation	114	114
Min	2	101
Max	14	1006
Mean	5.96	378.81
Median	5	325.50
Modus	4	311
Std	2.37	190.09
Range	12	905

4.2 Implementing linear regression

OLSR is implemented to get the CFP and the LOC conversion equation. In this regression model, several conditions must be met for the estimation model to be valid as an estimation tool. The linear regression model is called the BLUE if all these conditions are met.

Classical assumptions must be met in the OLSR model to become valid as a predictor. Classical assumptions in simple linear regression include interval or ratio data, linearity, normality, and homoscedasticity. Interval data provides quantitative data groups in numbers in which mathematical operations can be performed. The sequence between one data and other data has the same range, while the ratio data. In this study, both independent and dependent variables are ratio types.

Linearity is the nature of a linear relationship between variables, meaning that a change will follow every change occurring in one variable in the amount parallel to the other variables. The pearson correlation test uses the linearity test between the CFP and LOC variables. The test results are shown in Table 4 that the linearity assumption is fulfilled that the correlation value is 0.90 with p-value = $1.42 \cdot 10^{-42}$ (<0.05). Likewise, the normality test determines how the data is distributed. For samples between CFP and LOC variables, we use the Shapiro-Wilk test. The test results show that the logical value is one and p-value=$8.29 \cdot 10^{-98}$, which means the test successfully rejected the null hypothesis at the default 5% significance level. Homoscedasticity is a condition with similarities in the variance of errors for all observations of each independent variable in the regression model. For homoscedasticity tests, we use the Breusch-Pagan test; the p-value=14.31 is more significant than 0.05, so the null hypothesis cannot be dismissed from the model. Whereas for autocorrelation testing using Durbin Watson (DW), the DW value was 1.90 and p-value = 0.58, the number of variables k=1 and the number of

Table 4. The result of the test is to verify the assumption for OLSR

Test	value
Linearity test using the Pearson correlation statistic/p-value	$0.90/ 1.42 \cdot 10^{-42}$
Normality test using One-sample Kolmogorov-Smirnov statistic/ p-value	1(true)/ $8.29 \cdot 10^{-98}$
Homoscedasticity test using Breusch-Pagan statistic/ p-value	$14.31/1.55 \cdot 10^{-4}$
Autocorrelation test using Durbin Watson statistic/p-value	1.90/ 0.58

Table 5. The statistics about the OSLR model between CFP and LOC

	(Intercept)	CFP
Estimate	-51.23	72.2
SE	20.97	3.27
t-Stat	-2.44	22.06
p-Value	0.02	$1.42 \cdot 10^{-42}$
Mean Squared Error (MSE)	6820.09	-
R^2 Ordinary	0.81	-
R^2 Adjusted	0.81	-
F-stat	487	-

Table 6. The information about SVR models for each kernel functions

Kernel function	Linear	Polynomial	Gaussian
Number of Observation	114	114	114
Number of iterations	1,000,000	32,111	79
Epsilon	17.47	17.49	17.49
Beta	60.01	-	-
Bias	0.48	95.14	473.6
Order of function	1	3	-
MSE	7,969.14	6,702.91	6,214.66

Table 7. The function of conversion from CFP into LOC and x is CFP

Model	Function of conversion
OLSR	$f(x) = 72.20x - 51.23$
SVR-Linear	$f(x) = 60.01x + 0.48$
SVR-Polynomial	$f(x) = -0.37x^3 + 9.29x^2 + 2.65x + 95.14$
SVR-Gaussian	$f(x) = 731.99 * e^{-(\frac{x-13.47}{8.86})^2}$

observations n=114, from the DW table obtained dL=1.68 and dU=1.71 for significance level=5%. Because the DW value (1.90) is between dU to 4-dU, then there is no autocorrelation. The four tests' conclusions show that this study's dataset is BLUE.

Statistics about the OLSR model between CFP and LOC as independent and dependent variables are shown in Table 5. The OLSR model obtained is characterized by R^2=0.81, which shows that predictions are possible with a high confidence level, and the p-value is less than 0.05, indicating the build model's importance. Next, the plot graph for OLSR is given in Fig. 2, part (a).

4.3 Implementing support vector regression

As shown in section 3.2.2, SVR modelling uses three Kerner functions: Linear, Polynomial, and Gaussian. Information on SVR modelling for each kernel function is given in Table 6. Based on this table, the best MSE is SVR with Gaussian kernel function, followed by Polynomial and Linear kernels.

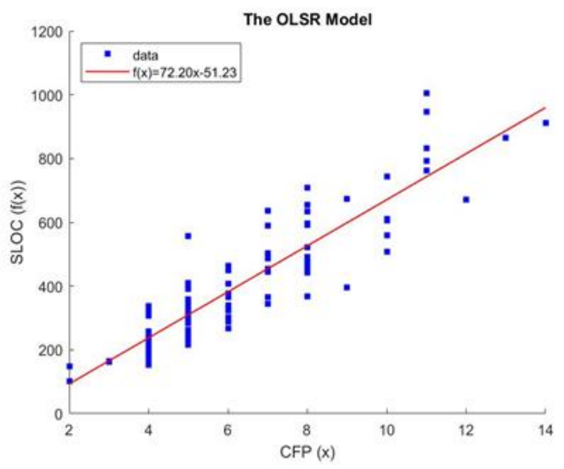
4.4 Conversion function

After modelling with OLSR and SVR, we got the results in Fig. 2. The figure shows the scatter and function curves for each model. The x-axis is CFP, and the y-axis is LOC. The function f(x) for each model is given in Table 7. Figs. 2 (a) and (b) are scatter plots for linear functions generated from OLSR and SVR-linear modelling, while Figs. 2 (c) and (d) are Polynomial and Gaussian functions.

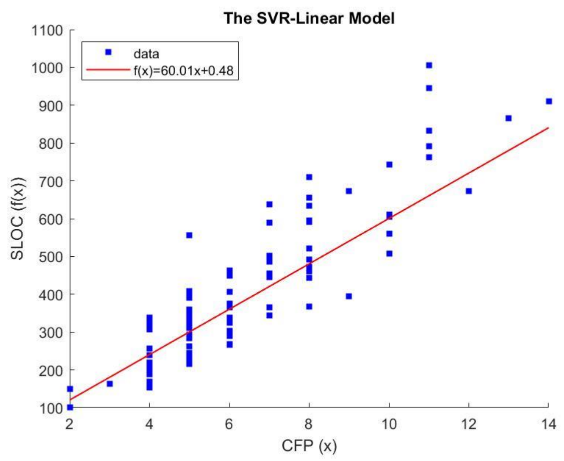
4.5 Comparison between UFP and CFP as main inputs on COCOMO II

We used four conversion functions to predict the LOC to test the accuracy of using CFP as the primary input to calculate effort in COCOMO II. We use the dataset obtained, namely the project ID=A, which took 30 function processes for testing. PM of COCOMO II in Eq. (1) is applied by giving the nominal values of E and EM (3 out of 5 on the linked scale) so that the values of E and EM are each valued at 1. Therefore, the PM formula becomes PM=A x Size. The constant A=2.94 and Size in kilos LOC, so the formula becomes PM=2.94 x size/1,000. Worksheet data for MRE, MMRE, and Pred calculations are given at https://intip.in/kymP.

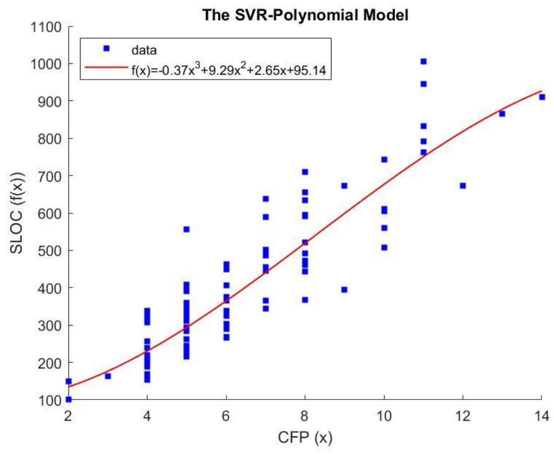
Test resumes are given in Table 8. The CFP size estimation model gives much better results than using UFP. The error rate of using CFP with MMRE is between 17%-19%, while using the UFP model has an error rate of 135%. Based on Pred (0.25), the CFP model confirms the MMRE measure with an accuracy rate of 67%-80%. While using UFP is only 10%. Meanwhile, comparing size estimation models using CFP shows that the OLSR model has the best accuracy based on MMRE and Pred (0.25) measures, namely 83% (or error 17%) and 80%.



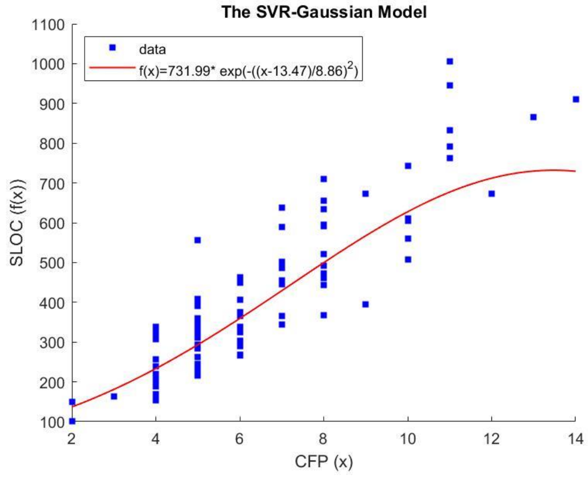
(a)



(b)



(c)



(d)

Figure. 2 The scatter and plot for four models, (a) Linear function of the OLSR model, (b) Linear function of the SVR-Linear model, (c) Polynomial function of the SVR-Polynomial model, and (d) Gaussian function of the SVR-Gaussian model

Table 8. The accuracy level of the four model

Measure	CFP Model (%)				UFP Model (%)
	OLSR	SVR Linear	SVR Polynomial	SVR Gaussian	
MMRE	17	19	17	18	135
Pred (0.25)	80	67	77	70	10

Table 9. Comparison with similar previous studies

Paper	Method	∑ instances in the dataset	Correlation	LOC of Language
[23]	Linear regression	14	0.417	C++
[11], [12]	Nonparametric statistics Spearman's rho	13	$Java_{LOC}$= 0.43, XML_{LOC}= 0.32/ $Java_{LOC}$= 0.89, XML_{LOC}= 0.872	Java and XML
[13]	linear regression	15	0.417	C++
[24]	Linear regression	25	0.8	Many languages
[3]	Not mentioned	-	-	-
This study	Linear regression and SVR	114	0.81	Java

4.6 Discussion

Two questions must be answered in this study. The first question is, 'How to convert from CFP to LOC?'. For this question, we have the answer that we have four conversion equations to get the LOC of CFP. The four models were obtained using OLSR, SVR with Linear kernel function, SVR with Polynomial kernel function, and SVR with Gaussian kernel function. The model was obtained using a dataset of 114 objects from the function process of four small-scale Java-based business applications.

The results of this study compared with previous studies on CFP to LOC conversion are given in Table 9. The amount of data used in this study is the largest, 114, compared to previous studies. The correlation between CFP and LOC variables in this study shows a strong correlation with a value of 0.81. Whereas in previous studies, it varied from having a weak correlation [11, 23] to a high [9, 10, 24]. As for the conversion to programming languages, in this study, specifically the conversion of LOC to Java Programming, the conversion results are helpful for software developer practitioners. The things above are the strengths of this study compared to similar previous studies.

The second question is, 'Is CFP as the main input of COCOMO II better than UFP?'. Our study found that the primary input of size in COCOMO II using the CFP model has a much better accuracy rate than the UFP model, as discussed in section 4.5. Therefore, CFP is worth considering as one of the main input measures in COCOMO II to complement the two previous methods that Boehm [2] proposed for the original COCOMO II. Therefore, input size can now use the previously mentioned methods and the CFP model.

Further, these results can be used by subsequent researchers or practitioners of software engineering, both programmers and project managers, to estimate the number of LOC generated during software project execution. For software engineering practitioners, the results of this study can be used study results as another way to estimate PM (or effort in person-months) in COCOMO II. For future researchers, these results can be further studied using datasets from applications other than business applications or in development environments based on programming languages other than Java.

5. Conclusion, limitations, and future study

The conclusions of the study are as follows:

1. In COCOMO II, the primary input for estimating the effort, duration, and cost of software development is the size of the software. Until now, in the original COCOMO II, size is obtained using the size stated in the LOC and Unadjusted Function Points (UFP). In this study, we considered another way to get the software size with the CFP model, in which CFP is the second generation of FSM. We have obtained four conversion functions from CFP to LOC, where LOC is used as the primary input for PM estimation in COCOMO II.
2. Accuracy testing using MMRE and Pred (0.25) for COCOMO II input showed that the CFP model (4 models) was much more accurate than the UFP model commonly used in the original COCOMO II. The CFP model has MMRE of 17%-19% and Pred (0.25) of 67%-80%, while the UFP model has MMRE and Pred (0.25) of 135% and 10%.
3. Among the four CFP models used for size as the primary input, the OLSR CFP model has the best accuracy based on MMRE and Pred (0.25), which is 17% and 80%.

Some of the limitations and future studies are given below.

1. The dataset used in this study is a small-scale business application software type. Therefore, it is necessary to conduct further studies to test its consistency for medium-large business application software projects.
2. The dataset used in this study is an application created using the desktop-based Java programming language; therefore, the results of this study need to be re-tested using web-based Java applications such as Servlet and JSP. For further studies, it is necessary to use a dataset consisting of Java web or Java mobile applications. It is also required to conduct further studies using datasets from other languages such as hypertext pre-processor (PHP). That programming language will produce a conversion ratio from CFP or function points to LOC or vice versa.
3. The dataset used in this study is business application software. Therefore, to generalize future studies using software types other than business applications, such as real-time applications, mathematically intensive applications, infrastructure software, and embedded systems.
4. The problem of data quantity also becomes a limitation when implemented using SVR, which requires extensive data to get optimal performance. Therefore, it is necessary to

conduct further studies using a dataset with more data to test the performance of the SVR model compared to the OLSR model.

Conflicts of interest

The authors (Sholiq Sholiq, Riyanarto Sarno, Endang Siti Astuti, and Muhammad Ainul Yaqin) declare no conflict of interest.

Author contributions

Conceptualization, Sholiq Sholiq and Riyanarto Sarno; methodology, Sholiq Sholiq; validation, Sholiq Sholiq; formal analysis, Sholiq Sholiq; investigation, Sholiq Sholiq and Muhammad Ainul Yaqin; dataset, Sholiq Sholiq; writing-original draft preparation, Sholiq Sholiq and Muhammad Ainul Yaqin; writing-review and editing, Riyanarto Sarno and Endang Siti Astuti; visualization Sholiq Sholiq; supervision, Riyanarto Sarno and Siti Endang Siti Astuti.

References

[1] C. Symons, A. Abran, C. Ebert, and F. Vogelezang, "Measurement of software size: advances made by the COSMIC community", In: *Proc. of 2016 Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement (IWSM-MENSURA)*, Berlin, Germany, pp. 75–86, 2016, doi: 10.1109/IWSM-Mensura.2016.021.

[2] B. Boehm, *Software Cost Estimation with COCOMO II*. New Jersey, USA: Prentice Hall, 2000.

[3] M. Haoues, A. Sellami, and H. B. Abdallah, "Towards functional change decision support based on COSMIC FSM method", *Information and Software Technology*, Vol. 110, pp. 78–91, Jun. 2019, doi: 10.1016/j.infsof.2019.02.004.

[4] H. Unlu, T. Hacaloglu, F. Buber, K. Berrak, O. Leblebici, and O. Demirors, "Utilization of Three Software Size Measures for Effort Estimation in Agile World: A Case Study", in *2022 48th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, Gran Canaria, Spain, pp. 239–246, 2022, doi: 10.1109/SEAA56994.2022.00045.

[5] T. Hacaloğlu, "Event Points: A Software Size Measurement Model", *Ph.D. - Doctoral Program*, Middle East Technical University, 2021. [Online]. Available: https://open.metu.edu.tr/handle/11511/93110

[6] A. Kaur and K. Kaur, "A COSMIC function points based test effort estimation model for mobile applications", *Journal of King Saud University - Computer and Information Sciences*, Vol. 34, No. 3, pp. 946–963, Mar. 2022, doi: 10.1016/j.jksuci.2019.03.001.

[7] A. L. A. L. Saleem and A. Y. Hammo, "Software Size Estimation: A survey", *Technium*, Vol. 4, No. 9, pp. 62–70, 2022, doi: 10.47577/technium.v4i9.7251.

[8] M. Haoues, A. Sellami, and H. B. Abdallah, "A rapid measurement procedure for sizing web and mobile applications based on COSMIC FSM method", In: *Proc. of the 27th International Workshop on Software Measurement and 12th International Conference on Software Process and Product Measurement*, Gothenburg Sweden, pp. 129–137, 2017, doi: 10.1145/3143434.3143436.

[9] C. Q. López, D. M. Sánchez, and M. Jenkins, "An Empirical Analysis of IFPUG FPA and COSMIC FFP Measurement Methods", In: *Information Technology and Systems: Proceedings of ICITS 2020*, pp. 265–274, 2020.

[10] A. Abran, J. Desharnais, and A. Lesterhuis, *The COSMIC Functional Size Measurement Method, Measurement Manual*, 2015.

[11] F. Ferrucci, C. Gravino, P. Salza, and F. Sarro, "Investigating Functional and Code Size Measures for Mobile Applications", In: *Proc. of 2015 41st Euromicro Conference on Software Engineering and Advanced Applications*, Madeira, Portugal, pp. 365–368, 2015, doi: 10.1109/SEAA.2015.23.

[12] F. Ferrucci, C. Gravino, P. Salza, and F. Sarro, "Investigating functional and code size measures for mobile applications: a replicated study", *Product-Focused Software Process Improvement*, P. Abrahamsson, L. Corral, M. Oivo, and B. Russo, Eds., Cham: Springer International Publishing, pp. 271–287, 2015, doi: 10.1007/978-3-319-26844-6_20.

[13] K. Lind and R. Heldal, "On the relationship between functional size and software code size", In: *Proc of the 2010 ICSE Workshop on Emerging Trends in Software Metrics*, Cape Town South Africa, pp. 47–52, 2010, doi: 10.1145/1809223.1809230.

[14] M. A. Sag and A. Tarhan, "Measuring COSMIC Software Size from Functional Execution Traces of Java Business Applications", In: *Proc. of 2014 Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement*, Rotterdam,

Netherlands, pp. 272–281, 2014, doi: 10.1109/IWSM.Mensura.2014.29.

[15] R. Gonultas and A. Tarhan, "Run-Time Calculation of COSMIC Functional Size via Automatic Installment of Measurement Code into Java Business Applications", In: *Proc. of 2015 41st Euromicro Conference on Software Engineering and Advanced Applications*, Madeira, Portugal, pp. 112–118, 2015, doi: 10.1109/SEAA.2015.30.

[16] A. Sahab and S. Trudel, "COSMIC Functional Size Automation of Java Web Applications Using the Spring MVC Framework", *IWSM-Mensura*, 2020.

[17] G. D. Vito, F. Ferrucci, and C. Gravino, "Design and automation of a COSMIC measurement procedure based on UML models", *Softw Syst Model*, Vol. 19, No. 1, pp. 171–198, 2020, doi: 10.1007/s10270-019-00731-2.

[18] Z. Kazi and L. Kazi, "Software Project Duration Estimation Based on COSMIC Method Applied to Data Flow Diagram", *IAJIT*, Vol. 19, No. 4, 2022, doi: 10.34028/iajit/19/4/8.

[19] Y. Attallah and H. Soubra, "Towards a COSMIC FSM Programming Language Compiler", *Presented at the CEUR Workshop Proceedings*, Izmir, Turkey, 2022.

[20] V. L. Martino and C. Gravino, "Using COSMIC to measure functional size of software: a Systematic Literature Review", In: *Proc. of 2022 48th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, Gran Canaria, pp. 225–228, 2022, doi: 10.1109/SEAA56994.2022.00042.

[21] A. Kaur and K. Kaur, "Systematic literature review of mobile application development and testing effort estimation", *Journal of King Saud University - Computer and Information Sciences*, Vol. 34, No. 2, pp. 1–15, 2022, doi: 10.1016/j.jksuci.2018.11.002.

[22] J. Wong, D. Ho, and L. F. Capretz, "A Neuro-Fuzzy Method to improving backfiring conversion ratios", *arXiv Preprint arXiv:1508.06191*, p. 6, 2015.

[23] C. Gencel, R. Heldal, and K. Lind, "On the Relationship between Different Size Measures in the Software Life Cycle", In: *Proc. of 2009 16th Asia-Pacific Software Engineering Conference*, Batu Ferringhi, p. 8, 2009, doi: 10.1109/APSEC.2009.51.

[24] S. D. Martino, F. Ferrucci, C. Gravino, and F. Sarro, "Assessing the effectiveness of approximate functional sizing approaches for effort estimation", *Information and Software Technology*, Vol. 123, p. 106308, 2020, doi: 10.1016/j.infsof.2020.106308.

[25] A. Kaur, "Comparative Analysis of Line of Code Metric Tools," *Int. J. Sci. Res. Sci. Eng. Technol.*, Vol. 2, pp. 1285–1288, 2016.

[26] O. A. M. López, A. M. López, and J. Crossa, "Support Vector Machines and Support Vector Regression", *Multivariate Statistical Machine Learning Methods for Genomic Prediction*, pp. 337–378, 2022.

[27] F. Zhang and L. J. O'Donnell, "Support vector regression", in *Machine Learning*, pp. 123–140, 2020, doi: 10.1016/B978-0-12-815739-8.00007-9.

[28] Z. Sakhrawi, A. Sellami, and N. Bouassida, "Support vector regression for enhancement effort prediction of Scrum projects from COSMIC functional size", *Innovations Syst Softw Eng*, Vol. 18, No. 1, pp. 137–153, 2022, doi: 10.1007/s11334-021-00420-8.

[29] A. J. Smola and B. Scholkopf, "A Tutorial on Support Vector Regression", *Statistics and Computing*, Vol. 14, pp. 199–222, 2004.

[30] M. Jørgensen, T. Halkjelsvik, and K. Liestøl, "When should we (not) use the mean magnitude of relative error (MMRE) as an error measure in software development effort estimation?", *Information and Software Technology*, Vol. 143, p. 106784, 2022, doi: 10.1016/j.infsof.2021.106784.

[31] H. H. Thai, P. Silhavy, M. Fajkus, Z. Prokopova, and R. Silhavy, "Propose-Specific Information Related to Prediction Level at x and Mean Magnitude of Relative Error: A Case Study of Software Effort Estimation", *Mathematics*, Vol. 10, No. 24, p. 4649, 2022, doi: 10.3390/math10244649.