Computing and Informatics, Vol. 44, 2025, 26–42, doi: 10.31577/cai_2025_1_26

EVALUATING COMBINED INFLUENCE OF WEIGHTED ANALYSIS CLASS DIAGRAM METRICS ON EARLY SOFTWARE SIZE ESTIMATION

Marriam Daud

Department of Software Engineering
National University of Modern Languages (NUML)
Islamabad, Pakistan
e-mail: marriamdaud@gmail.com

Ali Afzal Malik

FAST School of Computing
National University of Computer and Emerging Sciences (NUCES)
Lahore, Pakistan
e-mail: ali.afzal@nu.edu.pk

Abstract. Analysis class diagram (ACD) metrics like number of classes, number of methods, and number of attributes can be used for early software size estimation by project managers during initial project planning. However, not all of these ACD metrics have the same influence on software size. This study aims to empirically determine the relative influence of these ACD metrics on software size using historical data from academia and industry. Using the objective class points (OCP) metric as a base, two new metrics – enhanced OCP (EOCP) and weighted EOCP (WEOCP) – are proposed. Separate linear regression-based early software size estimation models are also constructed and validated using the original OCP metric and its newly proposed variants. A comparison of these models reveals that models based on our freshly proposed metrics perform better in terms of early size estimation accuracy.

Keywords: Analysis class diagram metrics, early software size estimation, information systems, objective class points, simple linear regression models, source lines of code, weights

1 INTRODUCTION

Software project plans are made around project schedules which, in turn, require estimates of task effort and duration. Estimates of task effort are highly dependent on the magnitude of the task and have the highest utility at the time of project inception. Therefore, accurate early estimation of software size in terms of source lines of code (SLOC) is of utmost importance for project managers in planning their software projects [1, 2].

Class diagram metrics have been widely used to predict software size (SLOC) in many existing studies [3, 4, 5, 6, 7, 8, 9, 10, 11]. A class diagram [12] illustrates the static structure of a software application. It depicts a set of classes (containing attributes and methods) connected to each other via different relationships. In the analysis stage of software development, an analysis class diagram (ACD) is created to understand the requirements of the problem domain. This paper focuses on exploiting metrics derived from an ACD to estimate SLOC early on in the project.

Objective class points (OCP) [13] is a metric that aggregates different class diagram metrics and has also been used by multiple earlier studies [6, 7, 10] to estimate software size in SLOC. However, these metrics are aggregated without considering their relative importance (via weights) in software size estimation. This research gap motivated us to determine the weights for these ACD metrics and investigate the combined influence of weighted ACD metrics on SLOC estimation.

In this paper, we propose two new metrics i.e. enhanced objective class points (EOCP) and weighted EOCP (WEOCP). EOCP is calculated by aggregating metrics for quantifying a relationship's multiplicity (one-to-one, one-to-many, and many-to-many) [12] into the original OCP. WEOCP is the weighted version of EOCP. For the computation of WEOCP, we utilize academic and industrial projects to empirically determine the weights for all the ACD metrics used in EOCP. The relative weights of these ACD metrics are determined using min-max normalization [14]. Subsequently, the WEOCP metric is computed utilizing these relative weights. Furthermore, we construct and validate regression-based SLOC estimation models based on the original OCP and its two proposed variants (i.e. EOCP and WEOCP). Results indicate that the models based on our proposed metrics (i.e. EOCP and WEOCP) outperform the original OCP-based model in terms of estimation accuracy.

The rest of this paper is structured as follows. Section 2 provides a summary and outlines the limitations of previous studies related to software size estimation using aggregated class diagram metrics. Section 3 describes our experimental design and Section 4 presents the experimental results of this study along with a discussion of the salient points. A worked-out example is provided in Section 5 to illustrate how our proposed model can be used by a project manager. Section 6 discusses the threats to the validity of this study. Finally, Section 7 summarizes the major conclusions and recommends future work.

2 SOFTWARE SIZE ESTIMATION USING AGGREGATED CLASS DIAGRAM METRICS

Costagliola et al. [15] proposed a software size estimation method called Class Points (CP). Their proposal included two variants i.e. CP1 and CP2. CP1 computation requires two metrics (i.e. number of services requested (NSR) and number of external methods (NEM)). CP2 is a refined version of CP1 since it involves three metrics (i.e. NSR, NEM, and number of attributes (NOA)). The results of empirical investigation for predicting development effort showed that the performance of both CP1 and CP2 was better than the single basic metrics used in CP counting. Results also showed that CP2 was better than CP1. However, the metrics involved in CP1 and CP2 counting were collected from design specifications that are available later in the software development.

A limited number of studies have evaluated the combined influence of class diagram metrics (e.g. number of classes, number of methods, etc.) in measuring or estimating software size. Kim et al. [13] proposed a metric called OCP that aggregates different class diagram metrics. However, OCP does not consider weights for different class diagram metrics even though these metrics represent different concepts.

Harizi [5] presented an approach to estimate software size using aggregated weighted class-level metrics. However, no objective approach was used to define the weights, no dataset was used to empirically validate the proposed method, and most importantly, the metrics from the design phase (rather than the earlier analysis phase) of the software development were used.

Zhou et al. [6] compared six different class diagram metrics-based SLOC estimation models (including the OCP-based model) that were built using eight different modeling techniques and two transformations. The best-performing model was based on an object-oriented project size metric that was built using ordinary least squares regression and logarithmic transformation. However, the class diagrams used in this study were collected by reverse-engineering the source code of the projects.

Badri et al. [7] conducted an empirical study to investigate the role of use case metrics and OCP in the early estimation of SLOC. They compared regression-based SLOC estimation models in terms of R^2. They found that the model utilizing use case metrics outperformed the OCP-based model. However, the collection of use case models and class diagrams involved a process of reverse-engineering the source code of the projects. Moreover, they used a small dataset and did not empirically validate these models.

In one of our earlier works [10], we empirically validated early SLOC estimation models based on OCP using six diverse datasets. In that work, however, different weights were not used for the different ACD metrics used in determining the value of the OCP metric.

To the best of our knowledge, the related literature does not report anything on empirically determining the weights for the ACD metrics aggregated in OCP and

on evaluating the combined influence of weighted ACD metrics on early software size estimation. This research attempts to fill this gap by empirically determining the weights for different ACD metrics (based on their contribution to software size) and by comparing the performance of different early SLOC estimation models built using the original OCP metric and its newly proposed variants.

3 EXPERIMENTAL DESIGN

The main goal of this research is to investigate and compare the estimation accuracy of different early software size estimation models built using OCP and its newly proposed variants i.e. EOCP and WEOCP. This goal is achieved by finding the answers to the following two research questions (RQs):

- RQ1: Is the EOCP-based model better than the original OCP-based model for early estimation of software size (SLOC)?
- RQ2: Is the WEOCP-based model better than the EOCP-based model for early estimation of software size (SLOC)?

3.1 Datasets

We collected 62 completed projects developed by students enrolled in two undergraduate level courses – object-oriented analysis and design (OOAD) and software engineering (SE) at a renowned private university located in Lahore, Pakistan. We also collected 11 completed projects developed by professionals at a private software development company located in Lahore. We assigned these projects to four different datasets based on their category (academia or industry), programming language, and project type (web or desktop), as shown in Table 1. These projects have also been used in our earlier works [10, 11].

Category	Dataset #	No. of Projects	Programming Language	Project Type
Academia	Dataset #1	31	C++	GUI-based desktop applications
Academia	Dataset #2	19	Java	GUI-based desktop applications
Academia	Dataset #3	12	Java	GUI-based web applications
Industry	Dataset #4	11	VB.NET	GUI-based desktop applications

Table 1. Datasets characteristics (adapted from [10])

3.2 Research Process

The steps to conduct experiments are illustrated in Figure 1 and listed as follows:

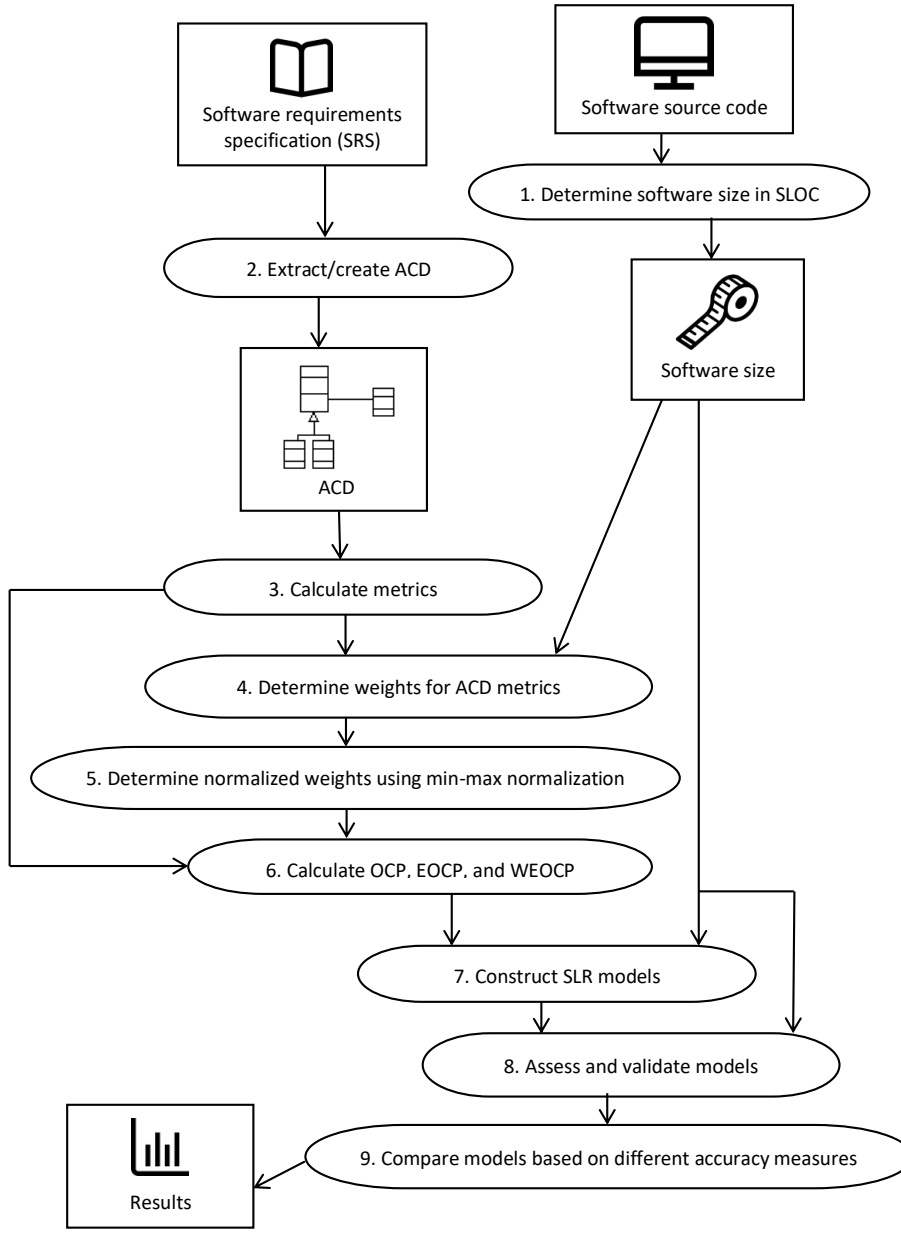

Figure 1. Steps of research methodology

1. Determine software size in SLOC from the source code of projects.
2. Extract/create ACDs from the software requirements specification (SRS) document of the projects.
3. Calculate ACD metrics from the extracted/created ACDs.
4. Determine weights for ACD metrics using simple linear regression (SLR).
5. Determine normalized weights for ACD metrics using min-max normalization.
6. Calculate OCP, EOCP, and WEOCP.
7. Build a separate regression-based SLOC estimation model based on each of the following metrics: OCP, EOCP, and WEOCP.
8. Assess and validate these models through the leave-one-out cross-validation (LOOCV) technique.
9. Compare these models with respect to different accuracy measures.

3.3 Metrics from UML Class Diagram

Table 2 lists the metrics that can be easily obtained from an ACD. We have proposed three new metrics – NOOM, NOMM, and NMMM (metrics 10–12 in Table 2) – that capture the information representing the multiplicity of relationships in a class diagram. Capturing this information is necessary since checks related to multiplicity need to be implemented explicitly in the source code thus requiring extra effort and consuming additional time. For example, to implement the one-to-one multiplicity between student and transcript, an extra check is required to make sure that no more than a single transcript object is associated with one student. Every student has a single transcript, and no student can have more than one transcript. EOCP (metric 14 in Table 2) is also a newly proposed metric. It is a variant of OCP which uses OCP as a base value and adds to it the information about multiplicities captured in NOOM, NOMM, and NMMM.

3.4 Weight Computation and Normalization

SLR [16] is a technique for predicting the value of a dependent variable (y) based on one predictor (x). Equation (1) presents the general form of an SLR model where "β_0" is the intercept of the line, "β_1" is the slope of the line, and ""ϵ" represents the error.

$$y = \beta_0 + \beta_1 x + \epsilon. \tag{1}$$

After removing outliers (8 from Dataset #1, 2 each from Dataset #2 and Dataset #3, and 1 from Dataset #4), SLR was used to calculate the R^2 value (response variable = SLOC, predictor = ACD metric) for each ACD metric using all four datasets. These R^2 values are shown in Table 3 which also shows the mean R^2 value for each ACD metric. We computed the mean R^2 value to determine the weights for all ACD metrics included in the definition of EOCP. The rationale for

Sr. #	Metric	Definition
1	NC	The total number of classes.
2	NA	The total number of attributes.
3	NM	The total number of methods.
4	NDep	The total number of dependency relationships.
5	NAss	The total number of association relationships.
6	NComp	The total number of composition relationships.
7	NAgg	The total number of aggregation relationships.
8	NGen	The total number of generalization relationships.
9	NRR	The total number of realization relationships.
10	NOOM	The total number of one-to-one multiplicity relationships.
11	NOMM	The total number of one-to-many multiplicity relationships.
12	NMMM	The total number of many-to-many multiplicity relationships.
13	Objective Class Points (OCP)	OCP = NC + NA + NM + NDep + NAss + NComp + NAgg + NGen + NRR
14	Enhanced Objective Class Points (EOCP)	EOCP = OCP + NOOM + NOMM + NMMM

Table 2. Metrics from UML class diagram

calculating the mean R^2 value despite the fact that these projects were developed in different programming languages is that the average gearing factors [17] for these programming languages (C++ = 50, Java = 53, and VB.NET = 52) are very close. Besides this, all of these projects belong to the same broad category i.e. GUI-based applications (see Table 1).

Min-max normalization was used to compute the normalized weights (see Table 3, last column) for all ACD metrics. This normalization technique rescales the weights to a new range of values such as $(0, 1)$. Equation (2), where Min is the minimum value of weight (0 in our data) and Max is the maximum value of weight (0.678 in our data), was used to determine the normalized weight for each ACD metric. Finally, these normalized weights were used to define the weighted version of the EOCP metric – weighted EOCP (WEOCP) – as shown in Equation (3).

$$Normalized\ Weight = \frac{Weight - Min}{Max - Min}, \tag{2}$$

$$\begin{aligned} WEOCP = {} & 1 * NC + 0.749 * NA + 0.889 * NM + 0.003 * NDep \\ & + 0.622 * NAss + 0.226 * NComp + 0.246 * NAgg \\ & + 0.777 * NGen + 0.243 * NOOM + 0.656 * NOMM \\ & + 0.385 * NMMM. \end{aligned} \tag{3}$$

Sr. #	ACD Metric	R^2				Mean R^2 (Weight)	Normalized Mean R^2 (NW)
		Dataset #1	Dataset #2	Dataset #3	Dataset #4		
1	NC	0.315	0.637	0.856	0.904	0.678	1
2	NA	0.024	0.734	0.444	0.831	0.508	0.749
3	NM	0.15	0.774	0.709	0.779	0.603	0.889
4	NDep	0.008	0	0	0	0.002	0.003
5	NAss	0.016	0.624	0.25	0.799	0.422	0.622
6	NComp	0.036	0.066	0.264	0.244	0.153	0.226
7	NAgg	0.001	0.011	0.136	0.519	0.167	0.246
8	NGen	0.502	0.651	0.48	0.475	0.527	0.777
9	NRR	0	0	0	0	0	0
10	NOOM	0.005	0.388	0.063	0.203	0.165	0.243
11	NOMM	0.013	0.325	0.63	0.811	0.445	0.656
12	NMMM	0.3	0.173	0.199	0.371	0.261	0.385

NW: normalized weight

Table 3. Weights for ACD metrics before and after normalization

3.5 Data Collection for Model Construction

SLR analysis was performed to construct three separate SLOC estimation models based on the original OCP and its two different variants (i.e. EOCP and WEOCP) as predictors. Values of these predictors were obtained using the information available in the ACD of completed projects. SLOC values, on the other hand, were automatically extracted from the source code of a project using a static analysis tool called Understand 5.1 [18]. Table 4 shows the descriptive statistics for all the variables used to construct the models.

3.6 Model Assessment Criteria

The three regression models are evaluated by using the most widely used accuracy metrics in the software size estimation community [10]. These include the mean absolute error (MAE) [10], the mean magnitude of relative error (MMRE) [10], the median magnitude of relative error (MdMRE) [10], Pred(25) [10], the sum of squared error (SSE) [12], and mean squared error (MSE) [10]. Equations (4), (5), (6), (7), (8) and (9) given below provide the formulas for calculating these performance measures where the subscript i represents an individual project, n is the total number of projects, and y_i and $\hat{y}_i$ are the actual and predicted sizes (SLOC), respectively, of project i.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|, \tag{4}$$

Dataset	Metric	Minimum	Median	Maximum	Mean	Std. Deviation
Dataset #1	OCP	36	91	175	94.94	31.28
	EOCP	38	100	198	104.87	35.14
	WEOCP	32	78	146	81.97	26.61
	SLOC	493	4 387	77 658	7 630.1	13 429.03
Dataset #2	OCP	41	90	308	111.79	66.207
	EOCP	47	100	362	127.26	78.45
	WEOCP	37	77	276	97.63	59.1
	SLOC	1 220	5 001	27 273	7 534	7 026.18
Dataset #3	OCP	17	67	98	65.42	25.34
	EOCP	18	73.5	109	72.5	28.81
	WEOCP	14	56.5	83	56.33	22.4
	SLOC	350	2 162	5 821	2 373.08	2 162.5
Dataset #4	OCP	526	911	1 409	951	233.92
	EOCP	573	1 000	1 517	1 028.27	249.88
	WEOCP	450	761	1 173	795	192.84
	SLOC	31 138	65 204	104 611	66 009.45	21 184.67

Table 4. Descriptive statistics for all metrics

$$MMRE = \frac{1}{n}\sum_{i=1}^{n} \frac{|y_i - \hat{y}_i|}{y_i}, \tag{5}$$

$$MdMRE = MEDIAN_{(i=1,\ n)}\left[\frac{|y_i - \hat{y}_i|}{y_i}\right], \tag{6}$$

$$Pred(25) = \frac{1}{n}\sum_{i=1}^{n} \begin{cases} 1, & \text{if } \frac{|y_i - \hat{y}_i|}{y_i} <= \frac{25}{100}, \\ 0, & \text{otherwise,} \end{cases} \tag{7}$$

$$SSE = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2, \tag{8}$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2. \tag{9}$$

4 EXPERIMENTAL RESULTS AND DISCUSSION

4.1 SLR Models

SLR analysis was performed using the IBM SPSS statistical tool [19, 20]. The rationale for using the SLR modeling approach is its simplicity, extensive usage, and documented higher effectiveness in improving estimation accuracy relative to alternative methods in the existing body of literature concerning software size estimation [6].

One of the important steps in SLR analysis is the identification of outliers (observations that fall vertically far from the regression line) and influential points (observations that fall horizontally far from the regression line) since these may significantly distort any SLR model [16]. Therefore, we identified and removed the outliers in each of the four datasets using both Cook's distance [21] and standardized residuals [10]. An observation with Cook's distance greater than $4/n$ was considered an influential point, where n is the total number of observations in a dataset [10, 11]. An observation was considered an outlier if its standardized residual value did not fall in the range of -2 to $+2$. We evaluated the quality of our linear models using the coefficient of determination (i.e. R^2) [16]. R^2 indicates the proportion of the variation in the response variable that is explained by the predictor.

Table 5 summarizes the results of applying SLR to our four datasets. For each dataset, the R^2 value of the best-performing model is highlighted. For Dataset #1, Dataset #2, and Dataset #3, the WEOCP-based model achieves the highest R^2 value. For Dataset #4, the EOCP-based model achieves the highest R^2 value. The linear relationship between these aggregated ACD metrics (OCP, EOCP, and WEOCP) and SLOC appears to be strong since most of the R^2 values are greater than or equal to 0.50. In Dataset #2, for instance, approximately 83 % of the variation in SLOC values can be explained by WEOCP (Model #3).

Dataset	Models (OCP, EOCP, WEOCP)	R^2	NI	Outlier(s)
Dataset #1	SLOC $= -2\,031.716 + 84.849 * \text{OCP}$	0.401		P1, P2, P4, P7, P10,
	SLOC $= -2\,489.104 + 83.363 * \text{EOCP}$	0.442	18	P14, P22, P24, P26,
	SLOC $= -2\,523.185 + 106.371 * \text{WEOCP}$	0.463		P27, P28, P29, P30
Dataset #2	SLOC $= -3\,651.555 + 97.911 * \text{OCP}$	0.824		
	SLOC $= -3\,720.238 + 88.290 * \text{EOCP}$	0.819	16	P14, P16, P47
	SLOC $= -3\,690.699 + 113.338 * \text{WEOCP}$	0.826		
Dataset #3	SLOC $= -525.847 + 39.391 * \text{OCP}$	0.728		
	SLOC $= -478.685 + 34.858 * \text{EOCP}$	0.736	11	P1
	SLOC $= -496.556 + 45.206 * \text{WEOCP}$	0.749		
Dataset #4	SLOC $= -3\,548.213 + 70.665 * \text{OCP}$	0.872		
	SLOC $= -4\,325.669 + 66.039 * \text{EOCP}$	0.877	10	P8
	SLOC $= -4\,661.618 + 85.953 * \text{WEOCP}$	0.875		

NI: Number of instances (i.e. projects) in a dataset after removing outlier(s)

Table 5. SLR results

The biggest improvement is seen in Dataset #1 in which the WEOCP-based model (Model #3) achieves an R^2 value which is about 15 % higher than that of the base OCP-based model (Model #1). This implies that our WEOCP-based model may be the most valuable for early size estimation of GUI-based desktop applications implemented in C++.

It is also observed that the EOCP-based model (Model #2) performs better than the original OCP-based model (Model #1) with respect to R^2 for 3 out of

4 datasets. The WEOCP-based model (Model #3) outperforms the original OCP-based model (Model #1) with respect to R^2 for all 4 datasets. On the other hand, the WEOCP-based model (Model #3) performs better than the EOCP-based model (Model #2) with respect to R^2 for 3 out of 4 datasets. Overall, at least one of the two newly proposed variants of OCP always performed better than the original OCP for all four datasets used in this study.

The WEOCP-based model (Model #3) is the best performing model for all three academic datasets (Datasets #1–3) while the EOCP-based model (Model #2) is the best performing model for the industrial dataset (Dataset #4). However, for the industrial dataset, the difference in the performance of our two newly proposed models is only marginal.

4.2 Model Assessment and Validation

Tables 6, 7, 8 and 9 show the values of the six accuracy measures (Section 3.6) for each of the three different models. These accuracy measures are obtained using the LOOCV method [22] separately on each of the four datasets. For each dataset, the accuracy metric value of the best-performing model is highlighted. For Dataset #1 (see Table 6), Model #3 is the best-performing model with respect to MAE, MdMRE, SSE, and MSE. With respect to MdMRE, Model #3 achieves around 16 % reduction vis-à-vis the base OCP-based model (Model #1). Model #2 performs better than all other competing models with respect to MMRE.

Accuracy	Model #1	Model #2	Model #3
Measure	SLOC ∼ OCP	SLOC ∼ EOCP	SLOC ∼ WEOCP
MAE	1 660	1 597	1568
MMRE	0.49	0.434	0.446
MdMRE	0.39	0.356	0.329
Pred(25)	0.222	0.222	0.222
SSE	60 845 099	56 601 137	54 488 670
MSE	3 380 283	3 144 508	3 027 148

Table 6. LOOCV results for Dataset #1

Accuracy	Model #1	Model #2	Model #3
Measure	SLOC ∼ OCP	SLOC ∼ EOCP	SLOC ∼ WEOCP
MAE	1673	1692	1613
MMRE	0.59	0.592	0.562
MdMRE	0.335	0.388	0.334
Pred(25)	0.25	0.312	0.312
SSE	62 113 178	63 671 263	61 270 333
MSE	3 882 074	3 979 454	3 829 396

Table 7. LOOCV results for Dataset #2

For Dataset #2 (see Table 7), Model #3 is the best-performing model with respect to all accuracy measures. For Dataset #3 (see Table 8), Model #3 is the best-performing model with respect to MAE, MMRE, SSE, and MSE. Model #2 performs better than all other competing models with respect to MdMRE. For Dataset #4 (see Table 9), Model #2 performs better than all the competing models with respect to MAE, MMRE, MdMRE, SSE, and MSE. Pred(25) values are equal for all three competing models for Dataset #1, Dataset #3, and Dataset #4. For Dataset #2, Model #2 and Model #3 outperform Model #1.

Overall, EOCP-based model (Model #2) performed better than the base OCP-based model (Model #1) in case of Dataset #1 (approximately 4 % reduction in MAE, 11 % reduction in MMRE, 9 % reduction in MdMRE, and 7 % reduction in SSE), Dataset #3 (approximately 2 % reduction in MAE, 2 % reduction in MMRE, 12 % reduction in MdMRE, and 3 % reduction in SSE), and Dataset #4 (approximately 2 % reduction in MAE, 3 % reduction in MMRE, 11 % reduction in MdMRE, and 4 % reduction in SSE). It is evident from these results that the prediction accuracy of the EOCP-based model (Model #2) is better than the original OCP-based model (Model #1) for most of the accuracy measures. This answers RQ1. WEOCP-based model (Model #3) performs better than the EOCP-based model (Model #2) for 3 out of 4 datasets for most of the accuracy measures. This answers RQ2.

Accuracy Measure	Model #1 SLOC ∼ OCP	Model #2 SLOC ∼ EOCP	Model #3 SLOC ∼ WEOCP
MAE	519	509	503
MMRE	0.346	0.34	0.333
MdMRE	0.23	0.202	0.215
Pred(25)	0.636	0.636	0.636
SSE	4 092 287	3 967 577	3 775 897
MSE	372 026	360 689	343 263

Table 8. LOOCV results for Dataset #3

Accuracy Measure	Model #1 SLOC ∼ OCP	Model #2 SLOC ∼ EOCP	Model #3 SLOC ∼ WEOCP
MAE	5 017	4 824	5032
MMRE	0.086	0.083	0.087
MdMRE	0.074	0.066	0.081
Pred(25)	1	1	1
SSE	365 546 841	351 053 261	356 142 761
MSE	36 554 684	35 105 326	35 614 276

Table 9. LOOCV results for Dataset #4

Table 10 summarizes the comparison of the performance (with respect to estimation accuracy) of all three models for all four datasets. This overall comparison

clearly reveals that both of the models based on the newly introduced variants of OCP (i.e. Model #2 and Model #3) lead to an improvement in the estimation accuracy vis-à-vis the original OCP-based model (i.e. Model #1).

Accuracy Measure	Dataset #1	Dataset #2	Dataset #3	Dataset #4
MAE	Model #3	Model #3	Model #3	Model #2
MMRE	Model #2	Model #3	Model #3	Model #2
MdMRE	Model #3	Model #3	Model #2	Model #2
Pred(25)	Equal	Model #3	Equal	Equal
SSE	Model #3	Model #3	Model #3	Model #2
MSE	Model #3	Model #3	Model #3	Model #2

Equal: All three models results are equal

Table 10. Best-performing model for all datasets

5 WORKED-OUT EXAMPLE

This section briefly explains how our proposed model based on WEOCP can be used by a practitioner (i.e. project manager) to estimate the size (in terms of SLOC) of a new software project.

Suppose a project manager has been assigned the responsibility of planning a new software project that requires developing a library management system (LMS). The requirement is to develop this GUI-based desktop application from scratch using the C++ programming language. The project manager receives the ACD of LMS from the business analyst. The project manager can now easily calculate the metrics from this ACD. Assume that the values of these metrics for LMS are those shown in Table 11, Step 1. The project manager can then calculate the WEOCP using Equation (3), as shown in Table 11, Step 2. Finally, this value (i.e. WEOCP = 85) can be plugged into the proposed WEOCP-based SLOC estimation model to estimate the SLOC of LMS (i.e. 6 518), as shown in Table 11, Step 3.

6 THREATS TO VALIDITY

In this section, we discuss the potential threats to the construct, internal, and external validity of this study, as well as the steps we have taken to address and mitigate these threats.

To address concerns about the construct validity of SLOC in the projects, we calculated SLOC using an automated tool called Understand. We selected this tool because it has been used in many existing software size estimation studies [6, 10, 11] for the same purpose. The independent variables were simple counts and manually calculated by the first author of this study based on the ACD available in the SRS of each project. To ensure correctness, these calculations were thoroughly reviewed by collaborators from academia and industry.

Step 1: Extract metrics from ACD
NC = 13, NA = 47, NM = 23, NDep = 0, NAss = 10,
NComp = 2, NAgg = 1, NGen = 3,
NOOM = 1, NOMM = 9, NMMM = 3
Step 2: Calculate WEOCP
$\text{WEOCP} = 1 * \text{NC} + 0.749 * \text{NA} + 0.889 * \text{NM} + 0.003 * \text{NDep} + 0.622 * \text{NAss} + 0.226 * \text{NComp} + 0.246 * \text{NAgg} + 0.777 * \text{NGen} + 0.243 * \text{NOOM} + 0.656 * \text{NOMM} + 0.385 * \text{NMMM}$
$\text{WEOCP} = 1 * 13 + 0.749 * 47 + 0.889 * 23 + 0.003 * 0 + 0.622 * 10 + 0.226 * 2 + 0.246 * 1 + 0.777 * 3 + 0.243 * 1 + 0.656 * 9 + 0.385 * 3$
WEOCP = 85
Step 3: Calculate estimated SLOC
$\text{SLOC} = -2\,523.185 + 106.371 * \text{WEOCP}$
$\text{SLOC} = -2\,523.185 + 106.371 * 85$
SLOC = 6 518

Table 11. Early software size (SLOC) estimation of a new software project

To reduce the threats related to the internal validity of this study, we used a reliable and robust model validation method, i.e. LOOCV. To improve the generalizability of our results, we examined and compared the performance of the proposed models using multiple accuracy measures and diverse datasets sourced from academia and industry. This approach helped alleviate concerns regarding the external validity of this study.

The models presented in this study may not be used as-it-is for all types of software applications. Nevertheless, we have presented a repeatable approach that practitioners can adopt to estimate software size in the early phase of object-oriented software development.

7 CONCLUSIONS AND FUTURE WORK

In this research, we have attempted to contribute to the domain of early software size estimation by exploiting a couple of unexplored gaps. None of the previous works has looked at the influence of different types of relationship multiplicities on software size. Similarly, while ACD metrics have been aggregated before to determine their influence on software size, the relative importance of these metrics (via weights) has not been studied so far.

This research study introduced a new metric called EOCP by aggregating the information about relationship multiplicities (present in an ACD) into the original OCP metric. Later, a weighted variant of EOCP (i.e. WEOCP) was also proposed by first empirically determining the weights for each of the different EOCP components and then normalizing these weights using min-max normalization. Furthermore, the study involved developing and validating separate regression-based SLOC estimation models based on the original OCP metric and the newly proposed EOCP and

WEOCP metrics. We then compared these models to determine their relative utility for early software size (SLOC) estimation. Results have shown that the models built using the newly proposed metrics (i.e. EOCP and WEOCP) perform better with respect to early software size estimation accuracy.

In the future, this study can be repeated using larger sets of real-life projects developed in different programming languages and environments. Furthermore, other normalization techniques may also be employed to determine weights for ACD metrics.

REFERENCES

[1] Galorath, D. D.—Evans, M. W.: Software Sizing, Estimation, and Risk Management: When Performance Is Measured Performance Improves. Auerbach Publications, 2006.

[2] Mahmood, Y.—Kama, N.—Azmi, A.—Khan, A. S.—Ali, M.: Software Effort Estimation Accuracy Prediction of Machine Learning Techniques: A Systematic Performance Evaluation. Software: Practice and Experience, Vol. 52, 2022, No. 1, pp. 39–65, doi: 10.1002/spe.3009.

[3] Mišić, V. B.—Tešić, D. N.: Estimation of Effort and Complexity: An Object-Oriented Case Study. Journal of Systems and Software, Vol. 41, 1998, No. 2, pp. 133–143, doi: 10.1016/S0164-1212(97)10014-0.

[4] Tan, H. B. K.—Zhao, Y.—Zhang, H.: Conceptual Data Model-Based Software Size Estimation for Information Systems. ACM Transactions on Software Engineering and Methodology (TOSEM), Vol. 19, 2009, No. 2, Art. No. 4, doi: 10.1145/1571629.1571630.

[5] Harizi, M.: The Role of Class Diagram in Estimating Software Size. International Journal of Computer Applications, Vol. 44, 2012, No. 5, pp. 31–33, doi: 10.5120/6261-8407.

[6] Zhou, Y.—Yang, Y.—Xu, B.—Leung, H.—Zhou, X.: Source Code Size Estimation Approaches for Object Oriented Systems from UML Class Diagrams: A Comparative Study. Information and Software Technology, Vol. 56, 2014, No. 2, pp. 220–237, doi: 10.1016/j.infsof.2013.09.003.

[7] Badri, M.—Badri, L.—Flageol, W.: Source and Test Code Size Prediction – A Comparison Between Use Case Metrics and Objective Class Points. Proceedings of the 11th International Conference on Evaluation of Novel Software Approaches to Software Engineering – Volume 1 (ENASE 2016), SciTePress, 2016, pp. 172–180, doi: 10.5220/0005857601720180.

[8] Prykhodko, S. B.—Shutko, I. S.—Prykhodko, A. S.: Early LOC Estimation of Web Apps Created Using Yii Framework by Nonlinear Regression Models. WSEAS Transactions on Computers, Vol. 20, 2021, pp. 321–328, doi: 10.37394/23205.2021.20.35.

[9] Prykhodko, S. B.—Shutko, I. S.—Prykhodko, A. S.: A Nonlinear Regression Model to Estimate the Size of Web Apps Created Using the CakePHP Framework.

Radio Electronics, Computer Science, Control, Vol. 2021, 2021, No. 4, pp. 129–139, doi: 10.15588/1607-3274-2021-4-12.

[10] Daud, M.—Malik, A. A.: Construction and Validation of Early Software Size Estimation Models Based on ADAF-Adjusted ACD Metrics. The Computer Journal, Vol. 66, 2023, No. 9, pp. 2123–2137, doi: 10.1093/comjnl/bxac065.

[11] Daud, M.—Malik, A. A.: Improving the Accuracy of Early Software Size Estimation Using Analysis-to-Design Adjustment Factors (ADAFs). IEEE Access, Vol. 9, 2021, pp. 81986–81999, doi: 10.1109/ACCESS.2021.3085752.

[12] About the Unified Modeling Language Specification, Version 2.5.1. OMG Unified Modeling Language, https://www.omg.org/spec/UML/2.5.1/PDF.

[13] Kim, S.—Lively, W. M.—Simmons, D. B.: An Effort Estimation by UML Points in Early Stage of Software Development. Software Engineering Research and Practice, CSREA Press, 2006, pp. 415–421, https://api.semanticscholar.org/CorpusID:17751474.

[14] Vafaei, N.—Ribeiro, R. A.—Camarinha-Matos, L. M.: Selection of Normalization Technique for Weighted Average Multi-Criteria Decision Making. In: Camarinha-Matos, L. M., Adu-Kankam, K. O., Julashokri, M. (Eds.): Technological Innovation for Resilient Systems (DoCEIS 2018). Springer, Cham, IFIP Advances in Information and Communication Technology, Vol. 521, 2018, pp. 43–52, doi: 10.1007/978-3-319-78574-5_4.

[15] Costagliola, G.—Ferrucci, F.—Tortora, G.—Vitiello, G.: Class Point: An Approach for the Size Estimation of Object-Oriented Systems. IEEE Transactions on Software Engineering, Vol. 31, 2005, No. 1, pp. 52–74, doi: 10.1109/TSE.2005.5.

[16] Wang, G. C. S.—Jain, C. L.: Regression Analysis Modeling & Forecasting. Graceway Publishing Company, 2003.

[17] Function Point Languages Table, Version 5.0. Quantitative Software Management (QSM), https://www.qsm.com/resources/function-point-languages-table.

[18] A Powerful Static Code Analysis Tool. Understand, https://scitools.com.

[19] IBM SPSS Modeler Statistical Tool. IBM SPSS Software, https://www.ibm.com/analytics/spss-statistics-software.

[20] Berkman, E. T.—Reise, S. P.: A Conceptual Guide to Statistics Using SPSS. SAGE Publications, 2012.

[21] Cook, R. D.: Detection of Influential Observation in Linear Regression. Technometrics, Vol. 19, 1977, No. 1, pp. 15–18, doi: 10.2307/1268249.

[22] Stone, M.: Cross-Validatory Choice and Assessment of Statistical Predictions. Journal of the Royal Statistical Society: Series B (Methodological), Vol. 36, 1974, No. 2, pp. 111–133, doi: 10.1111/j.2517-6161.1974.tb00994.x.

Marriam Daud is currently working as Assistant Professor at the Department of Software Engineering, National University of Modern Languages (NUML), Islamabad, Pakistan. She received her M.Sc. and her Ph.D. degrees in computer science from the National University of Computer and Emerging Sciences (FAST-NUCES), Lahore, in 2013 and 2022, respectively. She earned Gold Medal in BCS (Hons) with double majors (software engineering and information technology) from the Forman Christian College (FCC) (A Chartered University) in 2010. She also received Magna Cum Laude (Higher Honors) and a faculty outstanding student award in 2010 from FCC. She worked as a Senior Software Engineer at the WebCiters from 2009 to 2013. From 2013 to 2015, she worked as a Lecturer at the University of Lahore. Her research interests include software project management, software size and effort estimation, and database systems.

Ali Afzal Malik is currently working as Associate Professor of software engineering with the National University of Computer and Emerging Sciences (FAST-NUCES). After receiving the prestigious Fulbright scholarship in 2005, he obtained his M.Sc. and his Ph.D. degrees in computer science from the University of Southern California (USC), Los Angeles, USA in 2007 and 2010, respectively. He has undertaken research in software cost estimation at two of the world's leading research centers in software engineering, i.e. USC's Center for Systems and Software Engineering (CSSE) and the Institute of Software, Chinese Academy of Sciences (ISCAS). His current research work focuses on areas such as empirical software engineering and software cost estimation.