# A critical review of source code size estimation approaches for object-oriented programming languages: A comparative study

Majdi Ibrahim Alashhb
Belgrade Metropolitan University,
Majdi.Ibrahim.Alashhb.2588@metropolitan.ac.rs

LJUBOMIR LAZIĆ
Belgrade Metropolitan University,
Ljubomir.Lazic@metropolitan.ac.rs

*Abstract*—**Size estimation plays a key role in effort estimation that has a crucial impact on software projects in the software industry. We investigate the accuracy of early lines of code (LOC) estimation approaches for an object-oriented system at the early software development phase. We have validated the proposed methods using samples from both the software industry and open-source systems. In this paper, authors perform a comparative study of several size estimation models on C++, C#, and Java programs by using object-oriented metrics to estimate LOC, which comprise measures for #classes, class size, #methods per class, #attributes per class etc. We applied Surface Response Modeling (SRM) method, a case of Design of Experiment (DOE) method, in order to determine the factors that contribute most to the estimation model accuracy.**

***Key words - software size; LOC estimation; multiple regression analysis, SRM, DOE, estimation accuracy***

## I. INTRODUCTION

Size is an important internal attribute of software, which can be measured statically without having to execute the software system. Despite the fact that software sizing is well recognized as an important problem for more than two decades, there is still much problem in existing methods. A number of previous studies showed that size of C++, C#, and Java programs by using object-oriented metrics has strong correlations with many software characteristics such as the #classes, class size, #methods per class, #attributes per class [1-6], etc.

In spite of its importance, there have been relatively few empirical studies [1,2,4,6] published that investigate issues relating to the size estimation accuracy and confidence in LOC measure. We believe it is essential that we perform such a study to understand more about statistical importance of OO metrics used in software size estimation models for different OO programming languages such as C++, C#, and Java programs. By doing so, we hope we could achieve a better understanding of the OO metrics and their individual effects (contribution) to the software size and thus we could be able to more precisely estimate program size in LOC.

Many researchers propose a method for estimating LOC for an information system from its conceptual data model through the use of multiple linear regression models [2,4,5,8]. We have validated their methods through collecting samples from both the industry and open-source systems published in literature [1,3,4,5,6]. In this paper we report some experimental evaluations of metrics for OO models. Such metrics were obtained by adapting for OO models some metrics that ware originally conceived for OO code. According to the reported results, we sketch a methodological approach to the measurement of model size and other quantitative characteristics. A cross validation approach was adopted to build and evaluate linear and polynomial models where the independent variable was a traditional OO entity: classes, attributes, methods, association, inheritance, or a combination of them [4,9].

## II. RELATED WORK

### A. Parametric Modeling Process

The estimation of software size plays a key role in project effort estimation. Line of code (LOC) and function points (FPs) are the size measures most commonly used in existing software effort estimation models [10]. Despite the existence of well-known software sizing methods such as the function point method [4] and its variants tailored for object-oriented software [3,4,7], many practitioners and project managers continue to produce estimates based on ad hoc or so-called "expert" approaches. The main reasons cited are the lack of required information in the early stage of a project and the need of domain specific methods [5,7]. However, the accuracy of ad hoc and expert approaches has inherent problems that often results in upsets over project budgets and schedules [1,2,9,10]. It also affects the success of many projects.

Researchers have proposed several measures to assess different internal class attributes, such as size, cohesion, and coupling. The proposed OO size measures measure different program size aspects. For example, Number of Methods (NoM) measures the amount of functionality that a class provides and corresponding average size in LOC - AvgM, Number of Attributes (NoA) measures the amount of data necessary for the class to function and corresponding average size in LOC - AvgA size in LOC, and the lines of code (LOC) parameter measures the size in terms of statements [3,4,8,11] .

The entity-relationship (ER) model is widely used in conceptual modeling (requirements analysis) for data-

intensive systems. From our observation, the characteristic of a data-intensive system, and therefore the source code of its software, is well characterized by the ER diagram that models its data. Based on this observation, paper [4,6,8] proposes a method for building software size model from extended ER diagram through the use of regression models based on numbers of classes, methods, NoR – number of class relations (sum of associations, aggregations, inheritances). In the proposed approach, a separate software size model should be built for each different development environment (that is, each programming language and tool used). For example, different software size models should be built for systems written in Visual Basic with VB Script and SQL, and systems written in Java with JSP, Java Script and SQL.

However, we do not propose to build a software size model based on a fixed set of independent variables. It all depends on the kind of ER diagrams used in organizations for which we develop the software size model. Note that the above-mentioned association refers to association that is not aggregation. The separation of relationship types into associations, aggregations and generalizations is because of the differences in their semantics. These differences may result to some differences in navigation and updating needs in the database.

### *B. Approaches for Estimating Software Size in LOC*

A number of multiple regression models were developed with various combinations and transformations of the independent variables from collected metrics in a small dataset sample. Authors in work [3] proposed a linear combination of the number of methods (NoM) and the number of attributes (NoA) for LOC estimation (eLOC – without comments lines) based on 17 Java based projects (5 of them refer to the open source applications, while 7 refer to the small students' applications). In research [5] three simple linear models of size estimation that are based on class (AvgC) and method (AvgM) sizes in LOC. The parameters of the model have been experimentally derived using 25 applications written in Java. The designed model was experimentally verified using another 25 Java applications. The applications were totally independent from source ones, so authors assume that the results should apply to general cases. Authors in work [4] experimented with 29 subsystems, from four complete projects in the area of telecommunications, were measured. C++ was the target language, and the CORBA environment was the standard platform for distributed computing. Primary, the authors proposed object oriented function points (OOFP), an adaptation of the function points approach to object-oriented systems. In a small pilot study, authors used the OOFP method to estimate lines of code (LOC) and few simple linear models of size estimation that are based on NoC, NoM and NoR.

A cross validation approach was adopted to build and evaluate linear models where the independent variable was either a traditional OO entity (classes, methods, association, inheritance, or a combination of them) or an OOFP-related measure in our work [2].

Finally, the most related study to our work was undertaken by authors in work [4] who used OO metrics (NoC, NoR and AvgA=NoA/NoC) to build simple linear models of size estimation in LOC. From the dataset that is formed by the combination of the two datasets collected from the industry and open-source Java-based systems for model building, authors built end validated (Table 5,6,7 and 8 in article [4]) the simple linear models of size estimation in LOC.

## III. EMPIRICAL METHOD

The main finding of our previous work [1] is that *the effects of independent variables-metrics: NoC, NoM and NoA to the output variable LOC are NOT statistically significant.* Authors in works [3] and [4] did not statistically analyze their dataset in order to evaluate individual effects on LOC estimation confidence. This is the main reason why these authors [4] proposed and experimented with so many models (nineteen) which accuracy i.e. absolute relative error is in range of 25% to 107%. In this section we discuss the phenomenon based on deeper analysis of the data set from our project at MU student research on e-Learning tool [1] and 8 finished project data sets from literature review [3,4,5,6]. Perhaps the most valuable result of our study is not the set of LOC size estimation models, but a methodology able to consistently improve model performance described in this section. Indeed, during our validation, many factors were discovered that may affect the performance of the size prediction model.

### *A. Description of Data Sets*

As we already mentioned in previous section researchers have proposed several OO metrics (measures) to use in building prediction model for software Size in LOC such as: NoC, NoM, NoA, NoR, AvgA, AvgC and AvgM sizes in LOC. In order to compare several prediction models in literature we collected 9 datasets from multiple organizations in the industry including software house and end-users such as public organizations and insurance companies, open source systems published in articles [1,3,4,5,6]. These projects cover a wide range of application domains, implemented in various programming languages (C++, C#, Java etc.) including telecommunications, freight management, administrative and financial systems. The main objective for collecting data from open source systems is to have larger datasets. We used collected metrics of 16 applications (implemented in Java) produced by students in a Software Engineering course, denoted as Dataset_1 from work [3] which is similar to our students works [1] denoted as Dataset_3 consisting of 13 components written in C#. Because of dataset comparability, we used collected metrics of a large project (denoted as Dataset_2 from work [4]) consisting of 29 modules (components) written in C++ language. Denoted in this work as Dataset_4, consists of 25 applications from work [5, Table 3 and 5] which were used

to measure different Java characteristic metrics to build software site estimation model in LOC. Next, denoted as Dataset_5 is another independent 25 Java applications data set from work [5, Table 6] were used to validate generated size estimation model.

The last datasets, Dataset_6, 7, 8 and 9 are combination of 3) i) and 3) ii) datasets from Java based projects database presented in a research paper [6, Table 5,6,8 and 9]. Authors in [6] have validated their proposed estimation method through the following five pairs of datasets – each pair has one dataset for model building and another dataset for model validation – collected from the industry and open-source data-intensive information systems:

1) Industry VB-based System: These are collected from systems in the industry that were developed using Visual Basic with SQL. Both datasets have 16 systems.

2) Open-source PHP-based System: These are collected from open-source systems that were developed using PHP with HTML and SQL. The dataset for model building has 32 systems. The dataset for validation has 31 systems.

3) Java-based System: The following three pairs of datasets collected from systems that were developed using Java with JSP, HTML and SQL:

i) Industry Java-based System: These are collected from systems that were developed in the industry. Both datasets have 16 systems (our Dataset_6 and 7).

ii) Open-source Java-based System: These are collected from open-source systems. The dataset for model building has 30 systems (our Dataset_8). The dataset for validation has 24 systems (our Dataset_9).

iii) The combined Java-based System (our Dataset_10): These are formed by combing the corresponding datasets from the last two pairs of datasets. Therefore, the dataset for model building has 46 systems. The dataset for validation has 40 systems.

### *B. Building Size Models*

This research is driven towards achieving several objectives as follows:

• To analyze existing OO metrics, techniques and approaches used in building prediction model for software Size in LOC

• To formulate prediction model for software Size in LOC applying Surface Response Modeling (SRM) method, a case of Design of Experiment (DOE) method approach based on metrics in software OO design process at the project beginning phase (HLD)

• To evaluate the accuracy of proposed prediction model based on acceptable criteria for final selection of software Size in LOC prediction model.

This section describes the model-building strategies that were used for predicting software size in LOC.

The steps for building proposed software size models are as follows:

1) *Independent variables (OO metrics) identification*: Based on the type of data model (UML or ER diagram) constructed during requirements modeling and analysis.
2) *Data collection*: Collect UML and ER diagrams and sizes of source codes (in LOC) of sufficient data-intensive systems from published data in [1-6]. There are many free tools available for the automated extraction of source code size.
3) *Model building and evaluation*: There are quite a number of commonly used regression models. SRM of DOE regression models are considered in our research which is similar to works [1,4,6,8]. The size of source code (in LOC) and the independent variables identified in the first step form the dependent and the independent variables respectively for the model. Statistical packages (e.g. we used MINITAB ver.16 statistical software tool) should be used for the model building. Ideally, we should have separate datasets for modeling (model building) and evaluation. However, if the data is limited, the same dataset may also be used for model building and evaluation like in research works [3,4].

We applied Surface Response Modeling (SRM) method, a case of Design of Experiment (DOE) method (see Fig. 1, MINITAB 16 screenshots) in order to:

1. Fit a model to data collected using a central composite, Box-Behnken, or custom response surface design. We have chosen to fit the models with the following terms: *linear terms, squared terms and interaction terms (variables). We used for confidence level a commonly used α-level 0.05, then the p-value (P=0.05) to determine which of the effects in the model are statistically significant.* Before looking at the individual effects in the regression table, one should first look at the analysis of variance table at the p-values for the omnibus F-tests for all linear, all squared and all interaction effects. After identifying a significant set of effects (for example linear effects, or interaction effects), the regression table is to be used to evaluate the individual effects.
2. Find regression equation for Y as functions of independent variables $X_i$ in the form:

$$Y = b_0 + \sum_{i=1}^{n} b_i X_i + \sum_{i \neq j,1}^{n} b_{ij} X_i X_j \quad (1)$$

where : $Y$ - is the estimated (dependent) output variable i.e. LOC in our case, $n$ - number of independent variables-metrics used in the building model, $b_0$, $b_i$ – linear regression coefficients without variable interaction, $b_{ij}$ – variable interaction regression coefficients, and $X_i$ – real $i^{th}$ metrics values in the experiment.

### *C. Model assessment*

Further, the model building strategies have the following associated factors to compare software size estimation models characteristics of the goodness:

• The coefficient of determination, $R^2$ - it is the amount of variation explained by the regression equation which is used to predict future outcomes on the basis of other related information. It is a statistical term saying how good the

particular generated equation is at predicting functional defects. R-Sq. value must be above 50%.

• The F-ratio is used to test the hypothesis that all regression coefficients are zero at statistically significant levels.

Large deviations in size estimation area require model performance comparison using some heuristic rejection rules that compare more than just mean performance data [4]. The results for each validated method were compared using each treatment of MMRE, $SD_{MRE}$, and larger $R^2$ correlation. MMRE comes from the mean magnitude of the relative error, or MRE, the absolute value of the relative error:

$$MRE = |predicted - actual| / actual \quad (2)$$

The mean magnitude of the relative error, or MMRE, is the average percentage of the absolute values of the relative errors over an entire data set. Given *n* tests (estimators), the MMRE is:

$$MMRE = \frac{100}{n} \sum_{i}^{n} \frac{|predicted_i - actual_i|}{actual_i} \quad (3)$$

The standard deviation, or $SD_{MRE}$, is root mean square of MRE. Given a *n* tests (estimators), the $SD_{MRE}$ is:

$$SD_{MRE} = \frac{100}{T} \sum_{i}^{n} (MRE_i - MMRE)^{1/2} \quad (4)$$

The PRED(N) reports the average percentage of estimates that were within N% of the actual size candidates' estimated values in given T-test examples. MMRE and PRED(0.25) are the most commonly used assessments for validation. Lowest MMRE is preferred. The PRED(0.25) is the ratio of number of cases in which the estimates are within the 25% of the actual values divided by the total number of cases. The main criteria for validation lies in obtaining acceptable values of MMRE and PRED(0.25). Their acceptable values are not more than 0.25 and not less than 75% respectively according to [4].

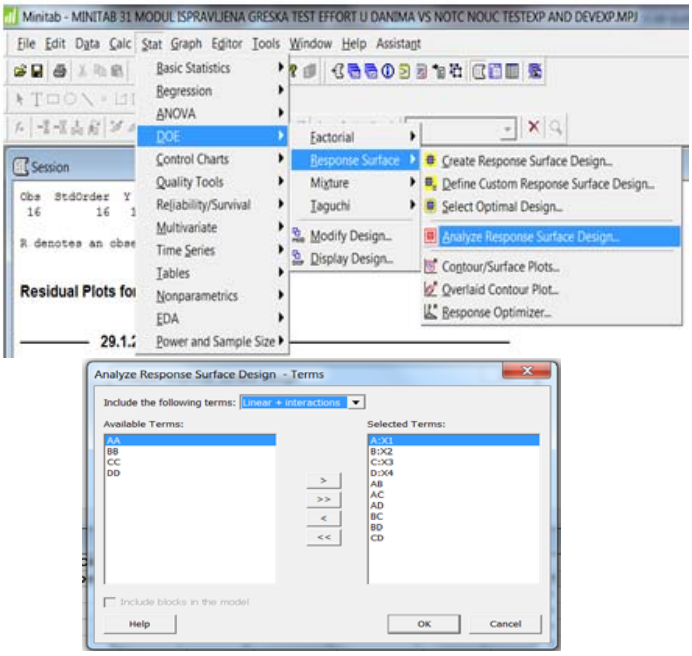
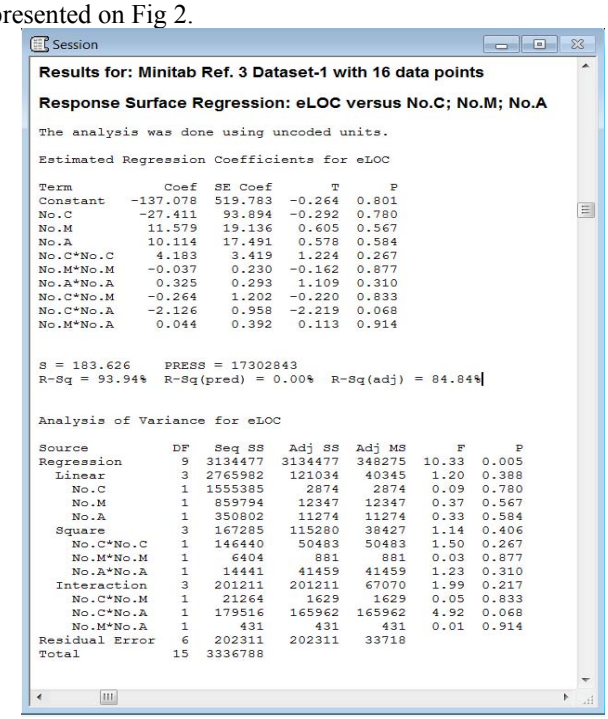



Figure 1. DOE->SRM MINITAB ver.16 statistical software screenshots

## IV. EMPIRICAL ANALYSIS OF STUDY

A report from MINITAB Response Surface Design to fit a SW Size estimation model to data collected using a central composite or Box-Behnken, on Dataset_1 from work [3] is presented on Fig 2.



Session

**Results for: Minitab Ref. 3 Dataset-1 with 16 data points**

**Response Surface Regression: eLOC versus No.C; No.M; No.A**

The analysis was done using uncoded units.

Estimated Regression Coefficients for eLOC

| Term | Coef | SE Coef | T | P |
|---|---|---|---|---|
| Constant | -137.078 | 519.783 | -0.264 | 0.801 |
| No.C | -27.411 | 93.894 | -0.292 | 0.780 |
| No.M | 11.579 | 19.136 | 0.605 | 0.567 |
| No.A | 10.114 | 17.491 | 0.578 | 0.584 |
| No.C*No.C | 4.183 | 3.419 | 1.224 | 0.267 |
| No.M*No.M | -0.037 | 0.230 | -0.162 | 0.877 |
| No.A*No.A | 0.325 | 0.293 | 1.109 | 0.310 |
| No.C*No.M | -0.264 | 1.202 | -0.220 | 0.833 |
| No.C*No.A | -2.126 | 0.958 | -2.219 | 0.068 |
| No.M*No.A | 0.044 | 0.392 | 0.113 | 0.914 |

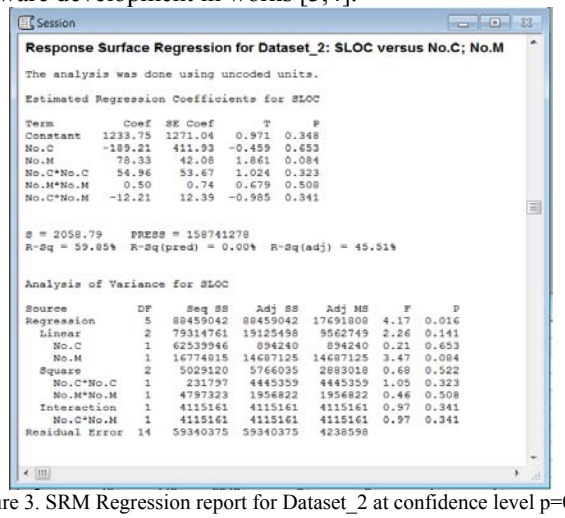
S = 183.626 PRESS = 17302843
R-Sq = 93.94% R-Sq(pred) = 0.00% R-Sq(adj) = 84.84%

Analysis of Variance for eLOC

| Source | DF | Seq SS | Adj SS | Adj MS | F | P |
|---|---|---|---|---|---|---|
| Regression | 9 | 3134477 | 3134477 | 348275 | 10.33 | 0.005 |
| Linear | 3 | 2765982 | 121034 | 40345 | 1.20 | 0.388 |
| No.C | 1 | 1555385 | 2874 | 2874 | 0.09 | 0.780 |
| No.M | 1 | 859794 | 12347 | 12347 | 0.37 | 0.567 |
| No.A | 1 | 350802 | 11274 | 11274 | 0.33 | 0.584 |
| Square | 3 | 167285 | 115280 | 38427 | 1.14 | 0.406 |
| No.C*No.C | 1 | 146440 | 50483 | 50483 | 1.50 | 0.267 |
| No.M*No.M | 1 | 6404 | 881 | 881 | 0.03 | 0.877 |
| No.A*No.A | 1 | 14441 | 41459 | 41459 | 1.23 | 0.310 |
| Interaction | 3 | 201211 | 201211 | 67070 | 1.99 | 0.217 |
| No.C*No.M | 1 | 21264 | 1629 | 1629 | 0.05 | 0.833 |
| No.C*No.A | 1 | 179516 | 165962 | 165962 | 4.92 | 0.068 |
| No.M*No.A | 1 | 431 | 431 | 431 | 0.01 | 0.914 |
| Residual Error | 6 | 202311 | 202311 | 33718 | | |
| Total | 15 | 3336788 | | | | |

Figure 2. SRM Regression report for Dataset_1 at confidence level p=0.05 from literature review [3]

The main finding of this work is that *the effects of independent variables-metrics:NoC, NoM and NoA to the output variable eLOC are NOT statistically significant.* We did the same analysis on collected metrics (NoC and NoM) of a large project (Dataset_2 from work [4]). The SRM Regression report at confidence level *p*=0.05 for Dataset_2 is presented in Fig. 3. The main finding in the previous text has shown that there is no significant correlation between the NoC, NoA and NoM in a model and the SW Size in software development in works [3,4].



Session

**Response Surface Regression for Dataset_2: SLOC versus No.C; No.M**

The analysis was done using uncoded units.

Estimated Regression Coefficients for SLOC

| Term | Coef | SE Coef | T | P |
|---|---|---|---|---|
| Constant | 1233.75 | 1271.04 | 0.971 | 0.348 |
| No.C | -189.21 | 411.93 | -0.459 | 0.653 |
| No.M | 78.33 | 42.08 | 1.861 | 0.084 |
| No.C*No.C | 54.96 | 53.67 | 1.024 | 0.323 |
| No.M*No.M | 0.50 | 0.74 | 0.679 | 0.508 |
| No.C*No.M | -12.21 | 12.39 | -0.985 | 0.341 |

S = 2058.79 PRESS = 158741278
R-Sq = 59.85% R-Sq(pred) = 0.00% R-Sq(adj) = 45.51%

Analysis of Variance for SLOC

| Source | DF | Seq SS | Adj SS | Adj MS | F | P |
|---|---|---|---|---|---|---|
| Regression | 5 | 88459042 | 88459042 | 17691808 | 4.17 | 0.016 |
| Linear | 2 | 79314761 | 19125498 | 9562749 | 2.26 | 0.141 |
| No.C | 1 | 62539946 | 894240 | 894240 | 0.21 | 0.653 |
| No.M | 1 | 16774815 | 14687125 | 14687125 | 3.47 | 0.084 |
| Square | 2 | 5029120 | 5766035 | 2883018 | 0.68 | 0.522 |
| No.C*No.C | 1 | 231797 | 4445359 | 4445359 | 1.05 | 0.323 |
| No.M*No.M | 1 | 4797323 | 1956822 | 1956822 | 0.46 | 0.508 |
| Interaction | 1 | 4115161 | 4115161 | 4115161 | 0.97 | 0.341 |
| No.C*No.M | 1 | 4115161 | 4115161 | 4115161 | 0.97 | 0.341 |
| Residual Error | 14 | 59340375 | 59340375 | 4238598 | | |

Figure 3. SRM Regression report for Dataset_2 at confidence level p=0.05 from literature review [4]

Authors in works [3] and [4] did not statistically analyze their dataset in order to evaluate individual effects on LOC estimation confidence. This is the main reason why these authors [4] proposed and experimented with so many models (nineteen) which accuracy i.e. absolute relative error is in range of 25% to 107%. In this section we discuss the phenomenon based on deeper analysis of the data from MU student research on e-Learning tool [1].

The analysis of variance table (Fig. 2 and 3) shows that by p-values less then 0.05 or close to this value for all linear, all squared and all interaction effects (for example linear effects, or interaction effects) on dependent output variable (eLOC) could only be metrics NoC and NoA. We apply SRM only on NoC and NoA metrics factors for our Dataset_3 of e-Learning application. Fig. 4 shows that statisticaly significant regression equation is [1]:

$$eLOC=22.361*NoA+0.269*NoA*NoA-2.109*NoA*NoC \quad (5)$$

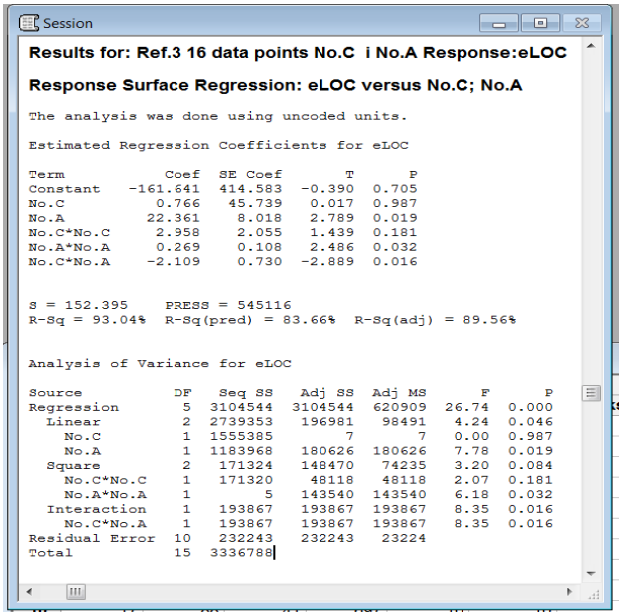



Figure 4. SRM Regression report on e-Learning dataset - Dataset_3 at confidence level p=0.05 [1]

In work [3] authors provided the formula:

$$eLOC = 3.3 * NoA + 5.7 * NoM \quad (6)$$

that computes eLOC with a reasonably small absolute relative error (23%). In our work [1] we analyzed eight estimation model candidates and our regression equation (5) computes eLOC with smaller absolute relative error. We can conclude from the estimation results that our regression equation (5) is better than estimation model (Eq. 6) presented in work [3], because MMRE is 36% compared to 68%. More importantly, 80% of e-Learning data points MRE<50%, while Eq. 6 provided only 8% with MRE<50%.

The SRM Regression report at confidence level $p$=0.05 for Dataset_4 from work [5] is presented in Fig. 5. Again, our approach showed that NoC and NoM have main contribution (significance) to Java application size in LOC, not only NoM multiplied by AvgM=12 LOC (average method size per method) as stated in work [5]. Our estimation equation is:

$$LOC=1913.89 -2.55312 * NoC + 11.1416 * NoM + 0.111281 * NoC * NoC \quad (7)$$

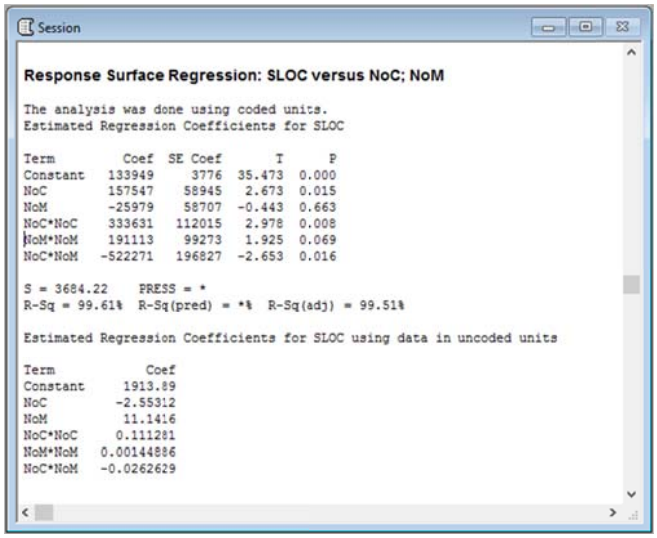



Figure 5. SRM Regression report on Java applications dataset - Dataset_4 at confidence level p=0.05 [5]

Estimating size of 25 Java application from Dataset_4 using equation (7) we obtained MMRE=17.4% instead 19.8% calculated by authors in work [5] (accuracy improvement by 2.5%) using Dataset_4 for model building. When we applied our more precise Eq. (7) on verification Dataset_6 from [5] we obtained MMRE=18% (small 0.6% deviation from building model dataset). It is surprising that proposed estimation formula in [5]:

$$LOC=12*NoM \quad (8)$$

provides better accuracy (MMRE=14.6%) than our model. It is well known [1,2,4,6,8] that estimation accuracy on validation dataset is always worse (MMRE greater) than on model building dataset. We applied estimation models from works: [3] (Eq. 6), works [5] (Eq. 8), and our model (Eq. 5) and calculated MMRE for our Dataset_3 from [1]. Results are presented on Fig. 6. It is confirmed that our SRM method provides better accuracy MMRE=35.3%, compared to MMRE=52.4% using model from [5] and MMRE=67.6% using model from [3]. We can conclude that authors in [5] made inversion and fitted application dataset.

| Sub.Sys Name | No.M | No.A | eLOC | Ref. [5] | MRE% | Ref. [3] | MRE% | Ref. [1] | MRE% |
|---|---|---|---|---|---|---|---|---|---|
| Form1 | 51 | 18 | 824 | 612 | 26% | 350 | 58% | 452 | 45% |
| FormPDF | 5 | 3 | 85 | 60 | 29% | 38 | 55% | 63 | 26% |
| FormKalkulatorR | 70 | 10 | 769 | 840 | 9% | 432 | 44% | 229 | 70% |
| FormInvOP | 12 | 13 | 353 | 144 | 59% | 111 | 68% | 309 | 13% |
| FormNeinvOP | 12 | 13 | 351 | 144 | 59% | 111 | 68% | 309 | 12% |
| FormDifOP | 12 | 15 | 392 | 144 | 63% | 118 | 70% | 364 | 7% |
| Form555bi | 19 | 26 | 668 | 228 | 66% | 194 | 71% | 708 | 6% |
| Form555mono | 17 | 15 | 636 | 204 | 68% | 146 | 77% | 364 | 43% |
| FormLogickaKola | 18 | 3 | 835 | 216 | 74% | 113 | 87% | 63 | 92% |
| FormLED | 13 | 15 | 356 | 156 | 56% | 124 | 65% | 364 | 2% |
| FormInduktivnost | 10 | 8 | 303 | 120 | 60% | 83 | 72% | 179 | 41% |
| FormTrafo | 10 | 14 | 317 | 120 | 62% | 103 | 67% | 336 | 6% |
| FormAutor | 2 | 0 | 47 | 24 | 49% | 11 | 76% | 0 | 100% |
| | | | | STD | 18.4% | | 10.3% | | 32.3% |
| | | | | MMRE | 52.4% | | 67.6% | | 35.6% |

Figure 6 Estimation accuracy of analysed models from [1,3,5].

When we applied our SRM proposed approach on Dataset_6, 7, 8, 9 and 10 *we obtained identical estimation equations* and confirmed that authors in work [6] correctly statistically analyzed used datasets. Research work in [6] is a confirmation of our approach i.e. SRM approach is verified (validated).

## V. CONCLUSIONS

It has been clearly demonstrated that Surface Response Modeling (SRM) analysis, a case of Design of Experiment (DOE) method has been successfully applied to formulate a software size in LOC prediction model building and validation. By using statistical approach, the research can justify the contribution and significance of OO metrics: NoC, NoM, NoA, NoR, AvgA, AvgC and AvgM, to experimentally derived software size prediction model precision using historical data base of finished projects in every company.

A number of multiple regression models were developed with various combinations and transformations of the independent variables. The models were then analyzed for their ability to accurately predict the dependent variable – software size in LOC.

In carrying out the research, the activities were subjected to several limitations. First, the research only produced parametric model design and validation, due to limited number of data points across application domains and implementation OO languages such as C++, C# and Java. Second, data collected is only limited to software development projects in which their metrics are rigorously collected and tracked.

By doing so, we hope we could achieve a better understanding of the OO metrics and their individual effects (contribution) to the software size and thus we could be able to more precisely estimate program size in LOC. We have validated the proposed methods using samples from both the software industry and open-source systems in order to determine the factors that contribute most to the estimation model accuracy.

Our SRM approach outperformed a few published research results on software size estimation in prediction accuracy on their project samples.

## ACKNOWLEDGEMENTS

This work has been done within the project 'Optimal Software Quality Management Framework', supported in part by the Ministry of Science and Technological Development of the Republic of Serbia under Project No.TR-35026.

## REFERENCES

[1] Lazić, Lj. and Milić S., Design of Experiments (DOE) methods in software engineering courses, eLearning 6th, Conference on e-Learning, 24 - 25 September / Belgrade, 2015.

[2] Lazić, Lj., et al., Comparative Study on Applicability of Four Software Size Estimation Models Based on Lines of Code, Proceedings of the 6th WSEAS EUROPEAN COMPUTING CONFERENCE (ECC'12), Prague, Szeh Republic, September 24-26, 2012. pp.71-80.

[3] V. del Bianco and Lavazza, L., Object-Oriented, Model Size Measurement: Experiences and a Proposal for a Process, Workshop on Model Size Metrics, The ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MoDELS 2006), Genova, October 2006.

[4] Antoniol, G. et al., Object-Oriented Function Points: An Empirical Validation, *Empirical Software Engineering*, 8, 225–254, 2003.

[5] J. Kaczmarek, M. Kucharski, Size and effort estimation for applications written in Java, *Information and Software Technology,* 46 ,2004, 589–601.

[6] H. Tan, Y. Zhao, H. Zhang. Estimating LOC for Information Systems from their Conceptual Data Models, ICSE'06, May 20–28, 2006, Shanghai, China, pp 321-330.

[7] Costagliola, G. et al., Class point: an approach for the size estimation of object-oriented systems, *IEEE Transactions on Software Engineering* 31 (1) (2005) 52–74.

[8] Y. Zhou, et al., Source code size estimation approaches for object-oriented systems from UML class diagrams: A comparative study, *Information and Software Technology* 56 (2014) 220–237.

[9] Boehm BW, Horowitz E, Madachy R, Reifer D, Clark BK, Steece B, Brown AW, Chulani S, Abts C. Software Cost Estimation with COCOMO II. Prentice Hall PTR: Upper Saddle River, NJ, 2000.

[10] S. H. Kan, "Metrics and Models in Software Quality Engineering," Second Edition, Addison-Wesley, 2003.

[11] J.Al Dallal, Object-oriented class maintainability prediction using internal quality attributes, *Information and Software Technology* 55 (2013) 2028–2048.