The Sixth International Conference on e-Learning (eLearning-2015), 22-23 September 2015, Belgrade, Serbia

DESIGN OF EXPERIMENT METHODS IN SOFTWARE ENGINEERING E-COURSES

LJUBOMIR LAZIĆ
Belgrade Metropolitan University, ljubomir.lazic@metropolitan.ac.rs

SLAVUJ MILIĆ
Belgrade Metropolitan University, slavuj.milic.1788@metropolitan.ac.rs

Abstract: *In this paper, we will tackle the pedagogical view of running empirical studies in software engineering (SE) e-courses. This is exemplified by several experiences in the teaching practice in order to show the benefits of empirical studies for students in the planning phase. As a result, diverse pedagogical benefits were obtained enhancing e-Learning platform. This paper focuses on improving SE student courses. The initial experiences on the usage of software metrics in the teaching of programming courses and concluding remarks are also presented. Among previously mentioned, one of the outcomes of the planning, execution and communication of the results, is the importance of an early training of students in the Design of Experiment (DOE) methods applied to few SE areas (size and effort estimation, testing and maintaining software products). This point of view is not strictly restricted to software engineering: good science in general is characterized by its emphasis on empiricism. This must be instilled into students whenever possible in software engineering courses. Running of the experiments in academic environments is a good way to achieve success in this respect. Along with the training in the empirical method, one must not forget that carrying out experiments with students educates them in topics that are close to state-of-the-art research.*

Keywords: *E-Learning, software engineering, empirical studies, training of students in the DOE methods*

1. INTRODUCTION

Software-intensive systems are increasing in importance due to their essential role in supporting everyday activities, as well as sustaining the global economy. Therefore, preparing software engineering students for technical design and decision making is critical. The evolution of software application domains toward web services, grid computing, intelligent software agents, and autonomic computing are leading to development of novel, agile, and flexible methodologies for software engineering.

Although most of the benefits that can be achieved by running experiments can also be acquired by other teaching techniques, these benefits are almost exclusively down to experiments. Furthermore, as a measurement of the benefits that this experience gave our students, we carried out a post-experiment session where we provided them with all the experimental material and the results. Afterwards, we discussed with the students the development and the outcome of the experiments by analyzing software size estimation, potential error number and estimation of repair costs [1]. The students were able to learn how to plan and perform experiments, analyse and interpret the empirical data. The experiments also allowed them to doubt statements that were widely accepted but have not been previously validated, when the results were not what they expected. These results demonstrate, that when planning empirical studies in software engineering courses one must keep in mind the pedagogical benefits they provide. A carefully planned experiment, conducted in the right teaching period and the right context (taking into consideration the skills and capabilities of the students of the course, topics, etc.) can benefit all the stake-holders involved.

The authors examine [1] the quality of the requirements, implementation, the project total effort, the process activity effort and the product size to assess the required time to finish student's project in SE courses. This time should not exceed 150 hours for each student project. In our case study, e-learning tool for basic electrical engineering course, we experimented with several software size estimation methods in order to assess required time to finish student's project.

The running of experiments in academic environments is a good way to achieve success in this respect. Along with training in the empirical method, we must not forget that carrying out experiments with students educates them in topics that are near to state-of-the-art research [3,7].

Initial experiences on usage of software metrics in teaching programming courses and concluding remarks are also presented. Among those previously mentioned, one of the outcomes of the planning, execution and communication of the results, is the importance of an early training of students in the Design of Experiment (DOE) methods applied to few SE areas (size and effort estimation, testing and maintaining software products).

2. E-LEARNING APPLICATION DESCRIPTION

The e-Learning application is primarily intended to be used as an assistance to students of secondary school, university students and electrical engineers in performing their daily tasks. The application itself should consist of the following [1]:

- Educational part
- Practical part

The educational part is to be comprised of **pdf** files containing: the most important laws of electrical engineering, indexing used in electrical engineering, elementary data about analog and digital circuits, their structure and use etc.

The practical part is to be comprised of various types of electrical calculators and simulators, by which the user should be able to perform required calculations in a relatively easy manner, and visually observe the functioning methods of the components and integrated circuits (through simulations).

The application should provide the following elements:

- Resistors (definition, types, values, resistance calculator, SMD resistors)
- Capacitors (definition, types, markings)
- Operational amplifier (ideal OA, inverting, non-inverting and differential amplifier),
- IC555 Circuit (characteristics, structure, monostable, bistable, examples of application),
- Logical circuits (AND, OR, NO, NOR, EXOR, EXNOR, circuit simulation),
- Principal laws of electrical engineering (Ohm's Law, Joule's law, Kirchhoff's laws, Coulomb's law)
- LED diode persistence
- Inductivity calculator
- Calculator of low power transformers

The conceptual model of the application is depicted in the following diagrams (image 1 to 8). Due to the specificity of the application and the organization of this work, only the basic diagrams are shown. The application is planned to consist of the main form, defined as " parent" that, through calls (by clicking on defined buttons), opens specific forms, defined as " children". Basically, the handling of the sub forms is repeated. The difference only exists between the elements contained in sub forms. Opening of all **pdf** files, on the other hand, is identical. A chosen **pdf** opens by clicking on a specified button.
The reason behind choosing this way to visualize a **pdf** file is accomplishing easy upgrading and/or change of the content in the application. By using this type of organization it is easy to add a new file, and a modification of an existing file comes down to memorizing a new pdf with the same name. Therefore, as creating a file is very simple to perform, it is also easy to make any changes.

Technologies and methodologies used for the creation of the application

The application is organized as a typical Windows desktop application, created in C# program language in **dotNet 4** environment. In its final version it is meant to be implemented as both, installation and portable version.
In order to improve its look and functionality, additional **Telerik** libraries were used (Image 1).

As for the structure of a class diagram in the application, it is organized through appropriate forms, each form having one class assigned to it. Changing of the way the forms are visualized is performed through Form1, which is a parent to all other forms. Each form is created as a separate class that opens within the main form that is superior to it.

In order to simplify, some classes and their methods and constructors will not be shown here.

After the application is started, the user can choose a part of the application he needs by using system menu.

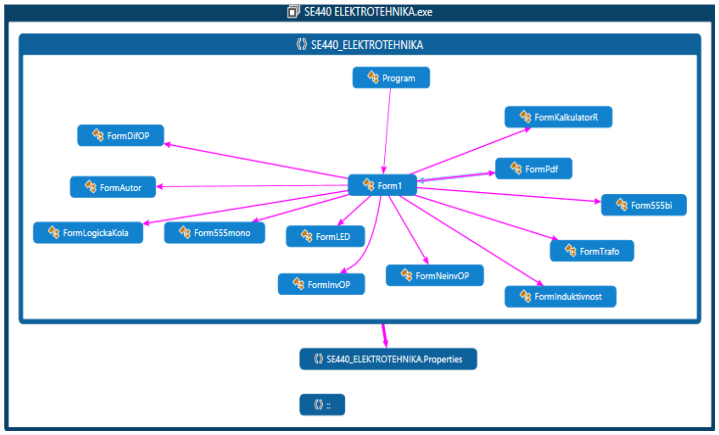


Image 1: The chart showing connections of classes within the application obtained by IDE Visual Studio tool

By clicking on a chosen button in the menu, the creating of the sub form starts (Image 2).

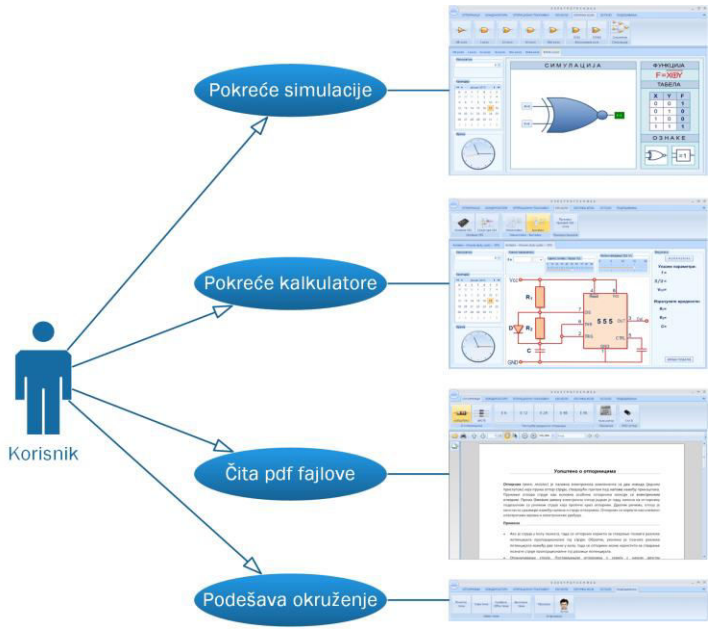


Image 2: User (Use Case) diagram

Use Cases

In general, the user can perform one of the following tasks:

1. Setting of the environment

2. Pdf file viewing
3. Starting the calculator in one of the electrical engineering disciplines

Starting of the simulations of electrical circuits
The user diagram is depicted in Image 2, showing the types of activities that can be performed by the user within the application.

The appearance of GUI application
To achieve better insight into how this application works, print screens of some of the forms are shown in the following images.

Image 3: Menu tab for resistors

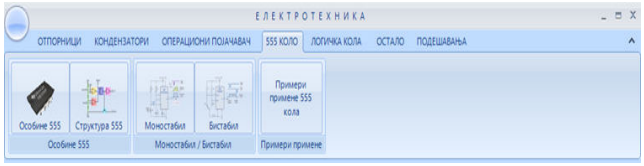


Image 4: Menu tab for 555 timer circuit

Image 5: Menu tab for logical circuits

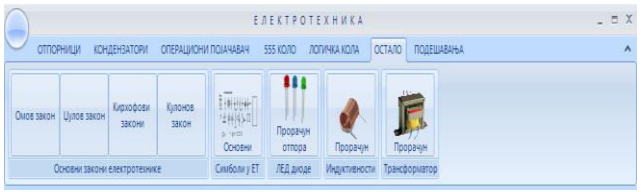


Image 6: Menu tab for the remaining parts of applications

The following images show the graphical interface of the application with a set theme:

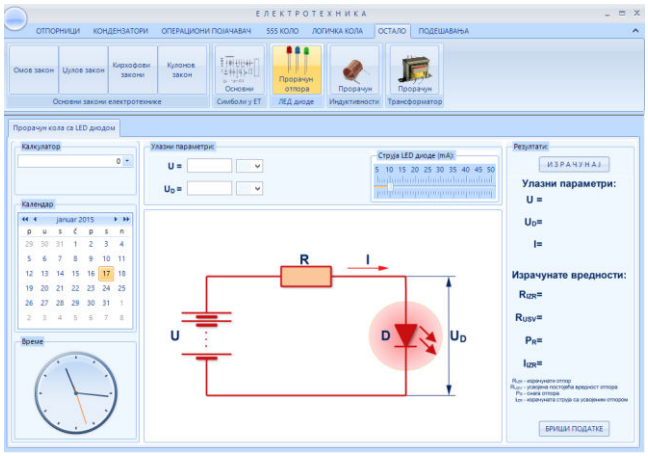


Image 7: Print screen of form of calculator for persistence of LED diodes

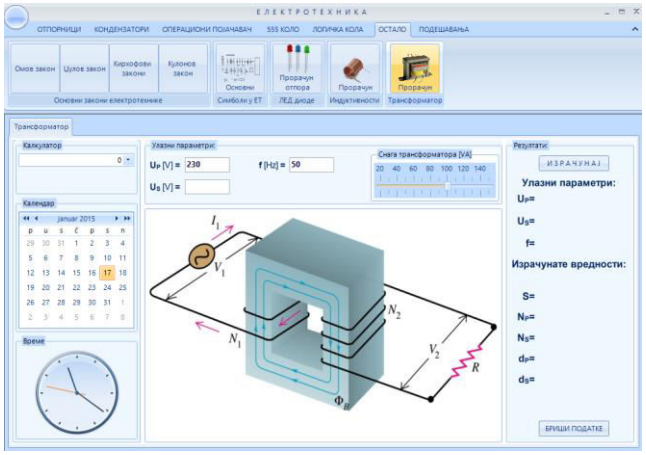


Image 8: Print screen of form of calculator for transformer elements

3. DOE IN E-LEARNING

Empirical studies in software engineering are essential for the validation of different methods, techniques, tools, etc. Students play a fundamental role in successful carrying out of these studies. As a consequence, most experiments connected with software engineering are conducted in Academia [1,3,7]. It is also very important to consider benefits of these studies from a teaching point of view. Therefore, when experiments are conducted in academia, they must be planned not only to obtain insights into research but also to help the students who participate as experimental subjects.

In many academic institutions in Serbia, the focus of engineering statistics is on the theory of probability (for example card shuffling, dice rolling, etc.), the mathematical aspects of probability and probability distributions (e.g. normal, exponential, binomial, Poisson, log-normal etc.), hypothesis testing etc. Quality improvement techniques (DOE, Taguchi's method, SRM, etc.) are often not covered. Understandably, graduates are not confident about using those techniques at their workplace.

As part of an exercise for increasing the awareness of DOE methods amongst software engineers, the authors used a collected students' project metrics (effort in hours to finish project, software size in SLOC, various metrics in OO design:#Classes, #Methods, #Attributes, #Forms, etc.) for the experiment at METROPOLITAN University (MU). Due to a limited amount of time, one member from each group in the class was assigned to do the experimental work. However, the students were all asked to analyse and interpret the software project data (on an individual basis). The results of the analysis were discussed in the classroom as part of the process of gaining an understanding experimental objectives and process.

Moreover, many courses do not cover the importance of careful experimental planning for the success of any industrially designed experiment.

The purpose of this experiment was to provide graduate engineering students with an understanding of the role of Surface Response Modeling (SRM) method, which is a case of Design of Experiment (DOE) method in MINITAB ver.16 statistical software tool. DOE is a series of ordered tests in which purposeful changes are made to input factors in order to identify the corresponding change in the output response variables. It is a statistical technique used to study the effect of the outcome of multiple variables simultaneously [6]. Software size estimation in our case study is output response variable and it depends on the metrics (input factors) used to find a formula for size estimation with required accuracy. This accuracy should be less than 50% at concept design phase (High Level Design - HLD) of e-Learning application described in previous section of this paper.

Empirical method

This section presents our empirical method of establishing software size in LOC (Size) and development Effort (E) estimation model for student's e-Learning application project. The two phases of the method are: (1) identifying software size and effort objectives to be managed quantitatively and constructing data samples for designing initial linear regression estimation model [3,4]; (2) establishing new Size and Estimation model according OO design process artifacts performance baseline for the identified metrics and cross validating all models using student's dataset [1].

Building SW Size and Effort Regression Models

This research is driven towards achieving several objectives as follows:

- analyzing existing metrics, techniques and approaches used for building prediction model for SW Size and E
- formulating prediction model for SW Size and E using statistical approach based on metrics in software OO design process at the project beginning phase (HLD)
- evaluating the accuracy of proposed prediction model based on acceptable criteria for final selection of SW Size and E prediction model.

This section describes our main research focus, the model-building strategies that were used for predicting SW Size and E. We built a mechanism for creating local models of effort estimation based upon size and other parameters, using variables tailored to the local environment applying following procedure.

Using the model to predict the effort for the new project:

1. Estimate the size
2. Use background equation to predict standard effort
3. Estimate the values of the factors used in the regression analysis
4. Compute the difference this project should exhibit based upon the values from 3. used in the multiple regression equation
5. Apply that difference to the standard effort to compute the improved effort estimate.

We adopted the following trade-off [2-5]:

- The coarse grain information concerning the structure of the system and the main elements (such as classes, functions, types etc.) are reported in detail in the intermediate representation. For instance, for every Java class we kept the complete information about the attributes and attribute types, methods and methods signatures, inheritance relations, etc.
- We summarized the information concerning the source code of finer grained elements (such as methods implementation) by computing various Source Lines of Code (SLOC) measures (total LOCs, effective eLOCs, #comment LOCs, etc.), the Cyclomatic Complexity [1], and the number and types of dependencies of a method.

The identification and interaction of these factors makes comparing productivity rates very difficult.

This is why software development databases should be statistically analyzed in order to determine the factors that contribute most to the specific database's productivity variation.

Once the variables—or combinations of variables—that explain most of the database's productivity variation are identified, the comparisons can be limited to the similar projects.

Multiple linear regression (MLR) analysis was used to model the relationship between quality, concept design and software metrics [1,5] based on two data samples: Dataset_1 and Dataset_2. In the first phase we used collected metrics of 16 applications produced by students in a Software Engineering course, (Dataset_1 from work [3]) which is similar to our students works. We also used collected metrics of a large project (Dataset_2 from work [4]) consisting of 29 modules (components) to preliminary build several candidates of SW Size and E regression model from work [4]. These authors [3,4,5] have extensive experience in the area of software metrics. They proposed sevral models to evaluate the characteristics of object-oriented design [3], to evaluate the characteristics of UML [4], estimate the development effort [5,6] etc. Based on that experience and after analyzing the literature on SW Size and Effort estimation, we selected three independent variables-metrics: X1 (Number of classes - NoC), X2 (Number of methods per class – NoM) and X3 (Number of attributes per class - NoA). In our experimental work, the dependent variable

that is to be measured is the eLOC (total lines of code without commenting lines) and LOC (total lines of code) i.e.Y [LOC]. The independent variables are the metrics collected by a student in e-Learning application using NDepend v.6 software tool (http://www.ndepend.com/) at HLD (concept), presented in Table 1. Table 1 also presented the measured LOC and eLOC, after coding phase, by which the estimation accuracy was calculated.

Table 1: e-Learning application OO metrics [1]

Class No.	Class name	Number of classes - NoC	Number of methods per class – NoM	Number of attributes per class - NoA	LOC (total)	eLOC (without commenting lines)
1	Form1	1	51	18	1218	824
2	FormPDF	1	5	3	106	85
3	FormKalkulatorR	1	70	10	1214	769
4	FormInvOP	1	12	13	468	353
5	FormNeinvOP	1	12	13	467	351
6	FormDifOP	1	12	15	516	392
7	Form555bi	1	19	26	866	668
8	Form555mono	1	17	15	827	636
9	FormLogickaKola	1	18	3	1141	835
10	FormLED	1	13	15	479	356
11	FormInduktivnost	1	10	8	409	303
12	FormTrafo	1	10	14	433	317
13	FormAutor	1	2	0	82	47
Total:					8226	5936

We applied Surface Response Modeling (SRM) method, a case of Design of Experiment (DOE) method in MINITAB ver.16 statistical software tool in order to:

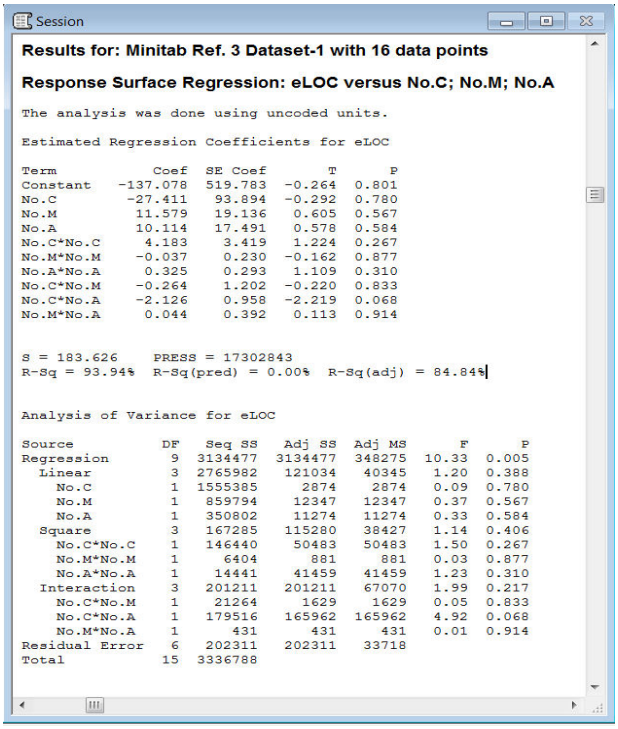
1. Analyze Response Surface Design to fit a model to data collected using a central composite, Box-Behnken, or custom response surface design. We have chosen to fit the models with the following terms: linear terms, squared terms and interaction terms (variables). We used for confidence level a commonly 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 X_1,X_2 and X_3 in the form:

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

where : Y - is the estimated (dependent) output variable i.e. SLOC (eLOC) i.e.Y [LOC] in our case, n - number of independent variables-metrics (3 in our case), 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.

4. EXPERIMENTAL ANALYSIS OF THE 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 Image 9.


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

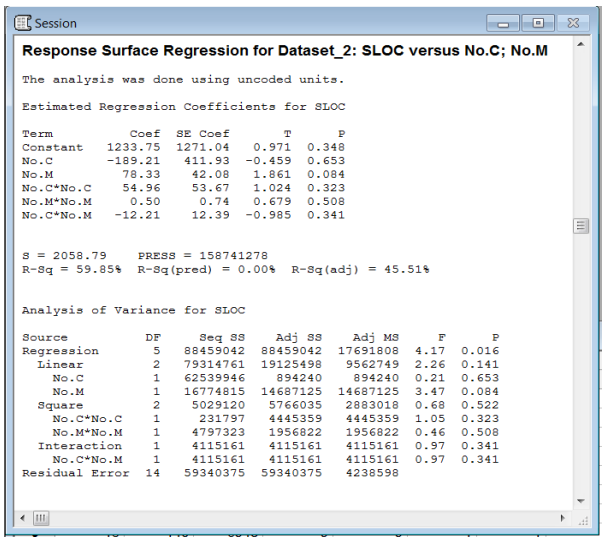
S = 183.626 PRESS = 17302843
R-Sq = 93.94% R-Sq(pred) = 0.00% R-Sq(adj) = 84.84%

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				

Image 9: SRM Regression report for Dataset_1 at confidence level α=0.05

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 α=0.05 for Dataset_2 is presented in Image 10. The main finding in the previous text has shown that there is no significant correlation between the number of class, number of attributes and number of methods 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		

Image 10: SRM Regression report on Dataset_2 at confidence level α=0.05

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 (Image 9 and 10) shows that by p-values less then 0.5 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 of e-Learning application. Image 11 shows that statisticaly significant regression equation is:

$$eLOC=22.361*NoA+0.269*NoA**2-2.109*NoA*NoC \quad (2)$$

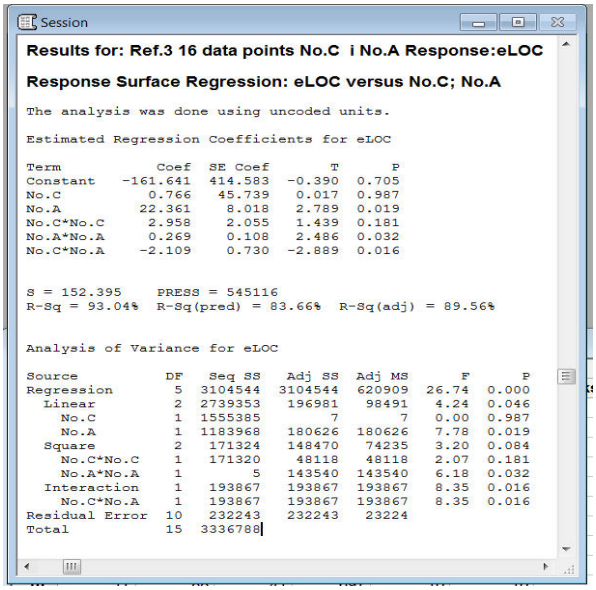


Image 11: SRM Regression report on e-Learning dataset at confidence level α=0.05

In work [3] authors provided the formula:

$$eLOC = 3.3 * NoA + 5.7 * NoM \quad (3)$$

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 (2) computes eLOC with smaller absolute relative error. Large deviations in size estimation area require model performance comparison using some heuristic rejection rules that compare more than just mean performance data [5]. In Table 2 we presented part of research results showing, besides absolute relative error: MRE (Magnitude Relative Error), MMRE (Mean Magnitude of the Relative Error) and SD (the Standard Deviation is root mean square of MRE).

Moreover, we considered another meaningful measure, the prediction at level N, defined as:

$$PRED(N) = \frac{100}{T}\sum_{i}^{T}\begin{cases}1 \cdot if \cdot MRE_i \le \frac{N}{100}\\ 0 \cdot otherwise\end{cases} \quad (4)$$

The PRED(N) reports the average percentage of estimates that were within N% of the actual size candidates' estimated values in given Dataset examples. MMRE and PRED(0.5), at the begining of the project (concept phase), are the most commonly used assessments for validation. Lowest MMRE is preferred.

Table 2: Results of e-Learning application Size estimation

Class No.	Class name	eLOC estimated by Eq. 2	MRE	eLOC estimated by Eq. 3	MRE	eLOC (measured)
1	Form1	452	45%	350	58%	824
2	FormPDF	63	26%	38	55%	85
3	FormKalkulatorR	229	70%	432	44%	769
4	FormInvOP	309	13%	111	68%	353
5	FormNeinvOP	309	12%	111	68%	351
6	FormDifOP	364	7%	118	70%	392
7	Form555bi	708	6%	194	71%	668
8	Form555mono	364	43%	146	77%	636
9	FormLogickaKola	63	92%	113	87%	835
10	FormLED	364	2%	124	65%	356
11	Forminduktivnost	179	41%	83	72%	303
12	FormTrafo	336	6%	103	67%	317
13	FormAutor	0	100%	11	76%	47
SDI:			32%		10%	
MMRE:		TotalEst: 3742	36%	TotalEst: 2988	68%	Total: 5936

We can conclude from the estimation results that our regression equation (2) is better than estimation mdel presented in work (3), because MMRE is 36% compared to 68%. More importantly, 80% of e-Learning data points MRE<50%, while Eq. 3 provided only 8% with MRE<50%.

Finally, after Size estimation we can estimate Effort in hours using formula of COCOMO Basic model [2]:

$$E=2.4*Size^{1.05}$$

where Size is in KLOC and E in months. Putting 0.3742KLOC for e-Learning application estimated Size, calculated effort is 0.85 month i.e. 0.85*152 hours (average hours per month) equals 128 hours, which is acceptable effort for student's project.

5. CONCLUSION

We have presented a student's e-Learning application and family of models for estimating the size in LOC of object oriented software once design artifacts are available at concept level.

We reported the empirical validation results. These results are very promising for size estimation, proposed in our empirical study. This is important, since an estimate of size is needed for many effort estimation models. In the data available to us so far, the best performance appears to be obtained with a model based on NoA, NoM and NoC, although its superiority is not statistically significant once compared with a model based on OO entities (classes, methods, inheritances and associations)

in cross validation process. Further experimentation is needed, with data from more systems, in order to statistically evaluate the difference, if any, among the proposed models.

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.

LITERATURE

[1] Milić, S., *Measurement and quality assessment: e-Learning application for electrical engineering*, Master thesis, METROPOLITAN Univesrsity, Belgrade, 2015.

[2] Tian, J.,*Software Quality Engineering, Testing, Quality Assurance, and Quantifiable Improvement*, John Wilay & Sons, Inc., Hoboken, New Jersey, 2005.

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

[4] Antoniol, G. et al., Antoniol atall, Object-Oriented Function Points: An Empirical,"Kluwer Academic Publishers, Netherlands, 2003.

[5] 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, ISBN: 978-1-61804-126-5, 2012. pp.71-80.

[6] Lazić Lj., and Milinković, S., Reducing Software Defects Removal Cost via Design of Experiments using Taguchi Approach, *Software Quality Journal*, Springer-Verlag New York, Inc., Volume 23, Issue 2, June 2015, pp. 267-295.

[7] Xinogalos S. and Ivanović, M., Enhancing Software Quality in Students' Programs, In: Z. Budimac (ed.): Proceedings of the 2nd Workshop of Software Quality Analysis, Monitoring, Improvement, and Applications (SQAMIA), Novi Sad, Serbia, 15.-17.9.2013.