Computing and Informatics, Vol. 43, 2024, 1234–1255, doi: 10.31577/cai_2024_5_1234

EXPLORING THE IMPACT OF SECURITY-BASED NON-FUNCTIONAL REQUIREMENTS ON EARLY SOFTWARE SIZE ESTIMATION

Marriam Daud

FAST School of Computing
National University of Computer and Emerging Sciences (NUCES)
Lahore, Pakistan
&
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. Software size – often measured in the source lines of code (SLOC) – fundamentally determines the software development effort. Realistic estimates of software size are, therefore, crucial for project planning. However, even though software size is influenced by both functional requirements (FRs) and non-functional requirements (NFRs), NFRs have been largely neglected in previous SLOC estimation studies. This study conducts an initial investigation of the impact of NFRs on early SLOC estimation by focusing on security-based NFRs related to data entry validations. First, the IFPUG software non-functional assessment process (SNAP) is used to calculate SNAP points for data entry validations (SPDEV). Then, SPDEV is used along with specially adjusted analysis class diagram (ACD) metrics to build and validate an early SLOC estimation model using an industrial dataset. Finally, the proposed model is compared with two existing size estimation models. Results

indicate that our proposed model outperforms both models in terms of estimation accuracy.

Keywords: Early software size estimation, information systems, IFPUG SNAP, linear regression, model validation, security non-functional requirements

1 INTRODUCTION

The importance of thorough planning in delivering software projects cannot be denied. Preparing a realistic and thorough project plan, however, depends largely extent on the availability of accurate estimates of project effort at the time of project inception. Accuracy of early effort estimates, in turn, relies on the accuracy of early estimates of software size, which is often measured using source lines of code (SLOC). This is precisely why software size estimates are a vital input of many algorithmic software efforts and cost estimation models [1, 2, 3, 4]. It may be reasoned that to reduce the chances of software failure, there is a compelling need for an accurate early SLOC estimation method.

Estimates of software size are primarily influenced by its functional requirements (FRs) and its non-functional requirements (NFRs). FRs and NFRs are the integral part of the software requirements specification (SRS) document for a software project. In ISO/IEC/IEEE 24765 [5], FR is defined as a requirement that describes what the software shall do to provide functional tasks and services to its users. On the other hand, NFR is defined as a requirement that describes not what the software will do for its users, but how the software will do it to provide functional tasks. NFRs include software quality attributes, technology constraints, design constraints, system constraints, etc [6].

One of the main reasons for the large difference between estimates of software size obtained at the project inception and measurements of actual size obtained at the project closure is not incorporating NFRs into early software size estimation [7]. In the existing literature, many methods [8, 9, 10, 11, 12] have been proposed for early SLOC estimation. However, to the best of our knowledge, none of these methods uses NFRs to estimate software size. This research attempts to fill in this gap. However, since this study conducts an initial exploration, its scope is limited to focusing on the influence of only one type of NFRs i.e. security-based NFRs related to data entry validations.

A lot of management information systems (MIS) use structured query language (SQL) for accessing and managing data in a relational database. SQL injection is one of the most serious security threats to these systems [13]. One of many causes of SQL injection attacks is the lack of data entry validations in these systems that allows attackers to enter malicious statements through input fields which, in turn, causes the execution of illegal queries resulting in unauthorized access to the database, data loss, misuse of sensitive data, and damage to the functionality and

confidentiality of the system [13]. Data entry validations are security-based NFRs that are often related to the FRs of the system. Software developers, thus, write SLOC for validation of input data, thereby increasing the size of the source code. This factor motivated us to incorporate security-based NFRs related to data entry validations into early SLOC estimation.

Daud and Malik [14] presented and empirically validated a new set of metrics called analysis-to-design adjustment factors (ADAFs). These metrics quantitatively capture the difference between the analysis class diagram (ACD) and the design class diagram (DCD) for four basic metrics i.e. number of classes (NOC), number of relationships (NOR), number of methods (NOM), and number of attributes (NOA). In the follow-up study [12], Daud and Malik used four ADAF-adjusted ACD metrics – ADAF-adjusted NOC (ANOC), ADAF-adjusted NOR (ANOR), ADAF-adjusted NOM (ANOM), and ADAF-adjusted NOA (ANOA) – to build and validate early SLOC estimation models. These models showed significant improvement in the estimation accuracy vis-à-vis to the existing relevant state-of-the-art SLOC estimation models.

In this study, we use software non-functional assessment process (SNAP) [15] sub-category 1.1 "data entry validations" to calculate SNAP points for data entry validations (SPDEV) for a dataset comprising 11 real-life industry MIS projects. We use SPDEV along with ADAF-adjusted ACD metrics to build and validate our proposed early SLOC estimation model. Moreover, we conduct a comparison of the estimation accuracy between the proposed model and two existing SLOC estimation models [12]. These existing models are constructed using the same dataset; however, they do not incorporate security-based NFRs pertaining to data entry validations. The results demonstrate that our proposed model exhibits significantly superior performance in terms of estimation accuracy, outperforming both existing models.

The remaining paper is structured as follows: Section 2 presents some previous work related to exploring and quantifying the relationship between NFRs and software size. Section 3 provides a brief summary of the prerequisite concepts necessary to understand this work. Section 4 elucidates the research methods utilized to carry out this study. Section 5 shows and discusses the results of model construction, assessment, validation, and comparison. Section 6 provides a practical demonstration, highlighting how practitioners can effectively apply our proposed model in real-world scenarios. Section 7 highlights the possible threats to the validity of this study. Finally, in Section 8, the main findings of the study are presented along with suggestions for future work.

2 NON-FUNCTIONAL REQUIREMENTS AND SOFTWARE SIZE

We have conducted a thorough literature review of English language conference papers or journal publications from 2005 to 2021. Our selection criteria focused on papers involving quantification methods, models, techniques, or approaches related to security-based NFRs or those based on quantifying NFRs using SNAP. The fol-

lowing paragraphs summarize the related research on quantifying NFRs and their impact on software size.

The earliest work on quantifying NFRs and measuring their impact on software size using functional size measurement (FSM) methods was reported by Kassab et al. [16, 17, 18]. Kassab et al. [17] quantified NFRs using the common software measurement international consortium (COSMIC) FSM method [19]. They illustrated the security, availability, and performance NFRs of a credit card system as operationalizing soft goals [19, 20] and then calculated the functional size in COSMIC function points (CFP) of these NFRs using the COSMIC FSM method. However, the usefulness of this technique is not empirically validated using real-life industry projects.

In another study, Kassab et al. [18] illustrated the quantification of performance requirements related to four basic functional processes of a mobile email system using the COSMIC FSM method. They discovered that after integrating performance requirements, the functional size increased by 117 %. However, this study lacks validation regarding the practical utility of the proposed approach through real-life industry projects.

Fellir et al. [21] proposed a hybrid technique that combines the COSMIC FSM method with case-based reasoning (CBR) [22] for including NFRs and their associations with FRs in the early estimation of software size and effort. However, no case study is employed to validate the effectiveness of the proposed technique.

Ochodek and Ozgok [23] investigated the relationships between the functional size measured by international function point users group (IFPUG) function point analysis (FPA) [24] and the non-functional size measured by SNAP [15] using three software applications. They discovered that the likelihood of a correlation between the non-functional size of transactional functions and their functional size is rather low.

Mansoor and Ochodek [25] used the SNAP to calculate the size of NFRs associated with the usability of user interfaces in web applications. They developed a prototype tool to calculate SNAP points (SP) for SNAP sub-category 2.1 "user interfaces" by taking input (i.e. HTML and CSS files) available later in the SDLC (after all the screens of an application are implemented). They tested the tool using only three web pages. However, this tool is sensitive to HTML and CSS errors and is not validated using real-life industry projects.

Ungan et al. [26] presented a method to measure the security-based quasi-NFRs by using FSM patterns [27] and the COSMIC FSM method. They applied FSM patterns to a small case study involving a course registration system and identified that the final functional size (after including security-based NFRs) is increased by 136 % as compared to the functional size measured in the initial phase of SDLC. However, only two basic functional processes are used to validate the proposed approach.

Meridji et al. [28] put forward a requirements model that relies on international standards to quantify security NFRs (i.e. availability, confidentiality, and integrity) using the COSMIC FSM method. Subsequently, in another study, Meridji

Sr. #	Author(s) (Year)	NFRs Quantification Method	Type of NFRs	Software Application Used	# DPs	Limitations
1	Kassab et al. (2007)	COSMIC FSM	Security, Availability, Performance	Credit Card System	1	No empirical validation through real-life industry projects.
2	Kassab et al. (2008)	COSMIC FSM	Performance	Mobile Email System	1	Only four functional processes are used to validate the proposed method.
3	Fellir et al. (2015)	COSMIC FSM and CBR	N/A	N/A	0	No empirical validation through the case study.
4	Ochodek and Ozgok (2015)	IFPUG SNAP	All in SNAP Categories and Sub-categories	Web Applications (2 Java, 1 PHP)	3	Small dataset
5	Mansoor and Ochodek (2016)	IFPUG SNAP sub-category 2.1 “user interfaces”	Usability of User Interfaces	Three web pages from yahoo.com, hitecmobile.sg, and github.com	1	Tool validation through only three web pages. Tool is sensitive to HTML and CSS errors.
6	Ungan et al. (2017)	FSM patterns and COSMIC FSM	Security	Course Registration System v. 2.0	1	Validation through only two functional processes.
7	Meridji et al. (2015)	COSMIC FSM	Availability, Confidentiality, Integrity	Automated Teller Machine	1	Validation through only one small case study.
8	Meridji et al. (2019)	COSMIC FSM	Availability, Confidentiality, Integrity	Automated Teller Machine	1	Validation through only one small case study.
9	Al-Sarayreh et al. (2021)	COSMIC FSM	Access Auditability, Confidentiality, Integrity, Data Encryption	Automated Teller Machine	1	Only the withdrawal process for an ATM system is experimented.

Table 1. Comparison of related work

et al. [29] introduced a framework for system security measurement based on standards. This framework enables the measurement of security NFRs such as availability, confidentiality, and integrity at the service level and the function level. They they illustrated measuring security requirements for an automated teller machine (ATM) system. However, only one small case study validates the proposed framework.

Al-Sarayreh et al. [30] presented a security framework for early identification and quantification of the functional and non-functional security requirements. They applied this framework to incorporate eight functional processes of an ATM system. The resulting total functional size was 41 CFP, comprising 30 CFP for the measurement of functional security measures and 11 CFP for non-functional security measures. However, the proposed framework is experimented with only using the withdrawal process for an ATM system. Table 1 summarizes and compares the related studies based on NFRs quantification methods, types of NFRs, number of projects (data points (DPs)) used to validate the research and their limitations.

3 PREREQUISITE CONCEPTS

The following sections briefly describe the three main concepts used in this research. A high-level understanding of these concepts is a prerequisite for grasping the content of this research.

3.1 SNAP Sub-Category 1.1 "Data Entry Validations"

SNAP is a method proposed by IFPUG to calculate the non-functional size of a software system using SNAP points (SP) [15]. It categorizes NFRs into 4 categories (Data operations, Interface design, Technical environment, and Architecture) and 14 sub-categories that are used to measure the non-functional size of a system. We use SNAP sub-category 1.1 "data entry validations" (that comes under Data operations) to calculate SPDEV.

The process to calculate the SPDEV of a software system has the following steps:

1. Analyze and identify NFRs related to sub-category 1.1 "data entry validations".
2. Identify SNAP counting units (SCUs). "The SCU is a component, a process or an activity, in which complexity and non-functional size are assessed" [15]. The SCU for SNAP sub-category 1.1 "data entry validations" is the elementary process (EP). EP is the smallest unit of activity that is self-contained, meaningful to the user, and represents a complete transaction [15]. For instance, "customer orders products" is an EP for an online retail store system.
3. Calculate SPDEV for each SCU considering two complexity parameters i.e. the number of data elements types (DETs) (unique and non-repeated fields used for validations) and the number of nesting levels (number of conditional validations

in the longest chain of validation). As shown in Table 2, first, the complexity (i.e. low, average, or high) of the SCU is identified based on the number of nesting levels. Then, SPDEV is calculated by multiplying a constant factor by the maximum number of DETs of that particular SCU.

4. Add the SPDEV of all SCUs to get the final SPDEV of the software system.

	Nesting Level Complexity		
	Low	Average	High
	1–2	3–5	6+
SPDEV =	$2 * \#$DETs	$3 * \#$DETs	$4 * \#$DETs

Table 2. SPDEV calculation (adapted from [15])

In this current research, we have used the aforementioned process to calculate the SPDEV values of all projects in our industrial dataset.

3.2 ADAF-Adjusted ACD Metrics

In this section, we summarize the steps for calculating ADAF-adjusted ACD metrics [14]. First, four basic metrics (i.e. NOC, NOR, NOM, and NOA) were calculated from the ACD of a project. Second, these same four metrics were automatically extracted from the DCD using a software tool called Understand [31]. Third, ADAF for each metric was calculated using the formula given in Equation (1), where values of x fall in the range of 1 to 4, representing the four basic metrics. DM is a DCD metric and AM is the same metric obtained from the ACD of a project i. Fourth, the mean ADAF (MADAF) for each metric was calculated using the formula given in Equation (2), where n represents the total number of instances (projects) in a dataset. Finally, each ACD metric (AM) was multiplied by the MADAF of that metric to obtain the ADAF-adjusted ACD metric (AAM) as shown in Equation (3).

$$ADAFx_i = \frac{DMx_i}{AMx_i}, \tag{1}$$

$$MADAFx = \frac{1}{n} \sum_{l=1}^{n} ADAFx_i, \tag{2}$$

$$AAMx_i = MADAFx * AMx_i. \tag{3}$$

The values of the four ADAF-adjusted ACD metrics (i.e. ANOC, ANOR, ANOM, and ANOA) for each project in the industrial dataset were determined in our earlier research [14]. These same values were used in this current research.

3.3 Objective Class Points

Kim et al. [32] presented a metric called objective class points (OCP) that can be easily calculated utilizing the information available in an ACD. It aggregates size metrics (i.e. NOC, NOM, and NOA) and structural complexity metrics (i.e. the number of generalization relationships (NOGen), the number of dependency relationships (NODep), the number of association relationships (NOAssoc), the number of aggregation relationships (NOAgg), the number of composition relationships (NOComp), and the number of realization relationships (NORR)). The formula for calculating OCP is given in Equation (4).

$$OCP = NOGen + NODep + NOAssoc + NOAgg + NOComp + NORR + NOM + NOC + NOA. \quad (4)$$

Like the values of the four ADAF-adjusted ACD metrics, the OCP values for projects in the industrial dataset were also determined in our earlier research [12]. These same values were used in this current research.

4 RESEARCH METHOD

We conducted a case study to assess the influence of the security-based NFRs (related to data entry validations) on early SLOC estimation using the steps described by the following sections. Sections 4.1–4.6 are based on the guidelines provided by Yin [33].

4.1 Research Questions

We answered the following research questions (RQs) for this study:

RQ1: How successful is our proposed model in accurately estimating software size measured in SLOC?

RQ2: Does incorporating security-based NFRs (related to the data entry validations) improve the estimation accuracy of the proposed model vis-à-vis the two existing early SLOC estimation models (i.e. ADAF-adjusted ACD metrics-based model and OCP-based model)?

4.2 Case and Units of Analysis

This research employed a single-case study design [33]. The case under investigation involved a well-established software house located in Lahore, which is a recognized member of the Pakistan software houses association for IT and ITeS (P@SHA) [34]. The units of analysis were 11 completed MIS projects [12, 14] implemented in VB.NET at the case software house. We selected these projects because these

projects belonged to the same category and were developed in the same environment. Moreover, we also had access to all the required artefacts (documentation and user manuals).

4.3 Data Collection

Daud and Malik [12, 14] determined the values of four ADAF-adjusted ACD metrics (ANOC, ANOR, ANOM, and ANOA), OCP, and software size (SLOC) for each project in the industrial dataset. The steps for calculating ADAF-adjusted ACD metrics are described in detail in Section 3.2. The SLOC of a project was automatically extracted using the tool Understand. In this current study, we calculated SPDEV (using NFRs available in the SRS document) for each project (Section 3.1). We used SPDEV along with ADAF-adjusted ACD metrics to build and validate an early SLOC estimation model. Table 3 shows the SPDEV, ANOC, ANOA, ANOM, ANOR, OCP, and software size (i.e. non-comment and non-blank physical SLOC) values for each project in our dataset. Table 4 illustrates the descriptive statistics for all metrics in our dataset.

Sr. #	Project Code	SPDEV	ANOC	ANOA	ANOM	ANOR	OCP	SLOC
1	p1	869	205	1404	1897	226	526	31 138
2	p2	1 127	250	2 894	2 101	343	839	46 085
3	p3	1 021	246	2 488	2 142	310	765	53 125
4	p4	1 269	262	2 581	2 703	320	842	66 481
5	p5	1 251	250	2 813	2 244	301	830	55 529
6	p6	1 271	275	2 975	2 448	428	911	67 448
7	p7	1 181	303	3 503	3 284	423	1 090	64 919
8	p8	1 253	308	3 695	3 764	390	1 164	104 611
9	p9	875	287	3265	3346	348	1 035	65 204
10	p10	1 379	308	3 428	2 856	489	1 050	73 075
11	p11	1 285	357	4 605	4 243	526	1 409	98 489

Table 3. Metrics' values for dataset (adapted from [12])

Metric	Minimum	Median	Maximum	Mean	Std. Deviation
SPDEV	869	1 251	1 379	1 161.91	170.72
ANOC	205	274.7	357	277.31	41.01
ANOA	1 404	2 975.4	4 605	3 059.24	809.94
ANOM	1 897	2 703	4 243	2 820.76	756.39
ANOR	226	347.8	526	373.01	88.31
OCP	526	911	1 409	951	233.92
SLOC	31 138	65 204	104 611	66 009.45	21 184.67

Table 4. Descriptive statistics for all metrics

4.4 Model Construction and Validation Methods

Pearson correlation coefficient method [35] investigates the degree and direction of linear association between two variables. The value of this coefficient lies between -1 and $+1$. For two variables, a value in close proximity to -1 indicates a near-perfect negative linear relationship, while a value near $+1$ indicates a near-perfect positive linear relationship between them. We used this coefficient to effectively identify the metrics that serve as significant predictors of SLOC.

We used linear regression[36, 37] to construct models for estimating SLOC of a software system. The rationale behind selecting this modeling method is that it is widely employed and has demonstrated superior performance compared to other methods in existing literature on software size estimation. [9, 12]. Linear regression involves two variables: dependent variable (a.k.a. response variable) and independent variable (a.k.a. predictor).

Simple linear regression (SLR) is used to predict the value of a dependent variable (y) using only one predictor (x). The formula for an SLR model is given in Equation (5).

$$y = \beta_0 + \beta_1 x + \epsilon. \tag{5}$$

Multiple linear regression (MLR) is used to predict the value of a dependent variable (y) using two or more predictors. The formula for an MLR model is given in Equation (6).

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_m x_m + \epsilon, \tag{6}$$

where "$x_1 \ldots x_m$" are predictors", "β_0" is the intercept of the line, "$\beta_1 \ldots \beta_m$" are regression coefficients, and "ϵ" represents the error. The quality of the linear models was assessed by using the coefficient of determination (R^2) [36]. It shows the percentage of variation in the dependent variable explained by the predictor(s). Stepwise MLR [12, 37] was used which builds a model by selecting the most suitable combination of statistically significant predictors based on the R^2 value.

Before applying statistical tests (i.e. Pearson correlation coefficients analysis and regression analysis), we made sure that the dataset had no outliers and that all variables followed the normal distribution. Cook's distance [38] was used to identify and remove outliers from our dataset. An instance in a dataset is identified as an outlier if its Cook's distance is greater than 4/n, where n is the total number of instances. We used the Shapiro-Wilk test [10] to examine the distribution of all variables. The null hypothesis for the Shapiro-Wilk test is that a variable is normally distributed in some population. We can reject the null hypothesis of population normality if the p-value of a variable is less than 0.05 (i.e. the variable is not normally distributed).

We used the leave-one-out cross-validation (LOOCV) technique [12, 39] to validate our proposed model. In this method, the numbers of iterations are equal to the number of instances in the dataset. It works by randomly choosing one instance as validation data and all the remaining instances as training data. Therefore, every instance in a dataset gets a chance to be used as validation data. For each iteration,

the errors are measured and, after the final iteration, the results are averaged. The rationale for using the LOOCV technique is that it provides a more robust, reliable, and unbiased estimate of model performance since each instance in a dataset can validate a model built using all other remaining instances. Moreover, it is the most widely used validation method in software size estimation studies [12].

In this research, we performed the statistical tests using the IBM SPSS statistical tool [40, 41]. The R programming language [42] was used to assess and validate the early software size estimation model.

4.5 Model Evaluation Criteria

To evaluate the estimation accuracy of the proposed early SLOC estimation model, we used Pred(25) (the number of predictions that are within 25 % of the actual) [12], mean magnitude of relative error (MMRE) [12], and median magnitude of relative error (MdMRE) [12] which are calculated based on magnitude of relative errors (MREs) (Equation (7)), where i denotes an instance (project) in a dataset, y_i denotes the actual size measured in SLOC, and $\hat{y}_i$ denotes the estimated size measured in SLOC. MMRE is recognized as an unreliable performance measure in the literature [42] since it is biased towards underestimation or overestimation due to the presence of outliers. To overcome this limitation, we used other accuracy measures – mean of absolute residuals (MAR) [12], sum of squared error (SSE) [12], and mean squared error (MSE) [12]. MAR is the average of the absolute differences between the actual SLOC and the predicted SLOC. SSE is calculated by adding together the squared differences between the actual SLOC and the predicted SLOC. MSE is calculated by taking the mean of the squared differences between the actual SLOC and the predicted SLOC.

$$MRE_i = \frac{|y_i - \hat{y}_i|}{y_i}. \tag{7}$$

4.6 Steps of the Proposed Method

The major steps of our proposed method are listed below:

1. Analyze the dataset for outliers and variable distribution.
2. Use Pearson correlation coefficients to investigate the relationship between SPDEV and software size (SLOC).
3. Construct an early SLOC estimation model using stepwise MLR based on SPDEV and ADAF-adjusted ACD metrics.
4. Validate the proposed model using the LOOCV technique.
5. Compare the proposed model with two existing early software size estimation models (i.e. ADAF-adjusted ACD metrics-based model and OCP-based model).

5 RESULTS AND DISCUSSION

This section presents the results obtained after executing each of the five major steps (Section 4.6) of our proposed method.

5.1 Dataset Analysis for Outliers and Variable Distribution

Cook's distance threshold for our dataset was $4/11 = 0.364$. Only one project's (p8's) Cook's distance (0.49548) was greater than the threshold value. Therefore, p8 was treated as an outlier and removed from our dataset. After removing the outlier, we used the Shapiro-Wilk test to examine the distribution of all metrics in our dataset. The p-values for SPDEV, ANOC, ANOA, ANOM, ANOR, OCP, and SLOC are 0.17, 0.869, 0.674, 0.334, 0.738, 0.674, and 0.635, respectively. Since all the metrics' p-values are greater than 0.05, we cannot reject the null hypothesis that all the metrics in our dataset are normally distributed.

5.2 Pearson Correlation Analysis

We studied the correlation between the predictors (SPDEV, ANOC, ANOA, ANOM, ANOR, and OCP) and the SLOC using the Pearson correlation method, as shown in Table 5. We found a significant positive correlation (p-value < 0.05) between all metrics and SLOC. Notably, SPDEV exhibited a strong and statistically significant correlation with SLOC, making it a promising candidate for predicting the SLOC of a software system.

	SPDEV	ANOC	ANOA	ANOM	ANOR	OCP
Pearson's r	0.613	0.951	0.912	0.883	0.878	0.934
p-value	0.03	< 0.001	< 0.001	< 0.001	< 0.001	< 0.001

Table 5. Pearson correlation between predictors and SLOC (adapted from [12])

5.3 Model Construction

First, SLR was used to investigate the predictive strength of each of the potential predictors in our dataset. Since outliers can significantly distort any regression model, we used Cook's distance to identify and remove an outlier from our dataset (Section 5.1). The results obtained from SLR are shown in Table 6. It is observed that SPDEV is the weakest standalone predictor of SLOC because its R^2 value is the lowest compared to other predictors.

Later, we used stepwise MLR to build a software size estimation model using SLOC as the dependent variable and SPDEV, ANOC, ANOA, ANOM, and ANOR as independent variables. We observed the proposed MLR-based model's statistics (i.e. R^2, F-test [12], t-test [12], signs of estimated coefficients, and variance inflation factor (VIF) [8, 43]) to evaluate its quality.

Sr. #	Predictors	R^2
1	SPDEV	0.376
2	ANOC	0.904
3	ANOA	0.831
4	ANOM	0.779
5	ANOR	0.77
6	OCP	0.872

Table 6. SLR results

Sr. #	Model	NI	R^2	ANOVA p-value	VIF	Outlier
1	$SLOC = -35\,820.07 + 18.93 * ANOM + 40.21 * SPDEV$	10	0.927	< 0.001	$ANOM = 1.082$ $SPDEV = 1.082$	p8
2	$SLOC = -48\,501.71 + 403.41 * ANOC$ [12]	10	0.904	< 0.001	$ANOC = 1.000$	p8
3	$SLOC = -3\,548.213 + 70.665 * OCP$ [12]	10	0.872	< 0.001	$OCP = 1.000$	p8

Table 7. Size estimation models before and after incorporating SPDEV

Table 7 lists the models obtained using stepwise MLR on our industrial dataset. Model #1 represents our proposed SLOC estimation model. The other two models – Model #2 and Model #3 – are from our earlier work [12]. Model #2 was constructed using only ADAF-adjusted ACD metrics (i.e. ANOC, ANOR, ANOM, and ANOA) as potential predictors. Model #3 was constructed using only OCP as a predictor.

The proposed model (Model #1) was built using SPDEV and ADAF-adjusted ACD metrics as potential predictors. It has the highest R^2 value (i.e. 0.927) which indicates that 92.7 % of the variation in SLOC can be explained by the predictors ANOM and SPDEV. This model is significant on the basis of the F-test since its analysis of variance (ANOVA) p-value (i.e. 0.000103) is much less than the significance level ($\alpha = 0.05$). The p-values of ANOM and SPDEV are 0.000164 and 0.007, respectively. Since the p-values of both predictors are less than the significance level ($\alpha = 0.05$), this implies that both predictors are significant and the proposed linear model is a good fit to the data according to the t-test. VIF values for the predictors of this model are close to 1 which shows that multicollinearity between the predictors is very low. The positive estimated coefficients of ANOM and SPDEV show that software size (SLOC) is expected to increase when the ANOM and SPDEV increase. More specifically, the value of SLOC is expected to increase by almost 19 units for a unit of increase in ANOM and by almost 40 units for a unit of increase in SPDEV.

Figure 1 depicts information about the standardized residuals for our proposed model. An essential assumption of the regression method is that the residuals are normally distributed (i.e. 95 % of values of standardized residuals lie between −3 and +3) [38]. The standardized residuals for our proposed model are normally distributed (Figure 1). Figure 2 shows the relationship between actual SLOC and estimated SLOC captured using our proposed model.

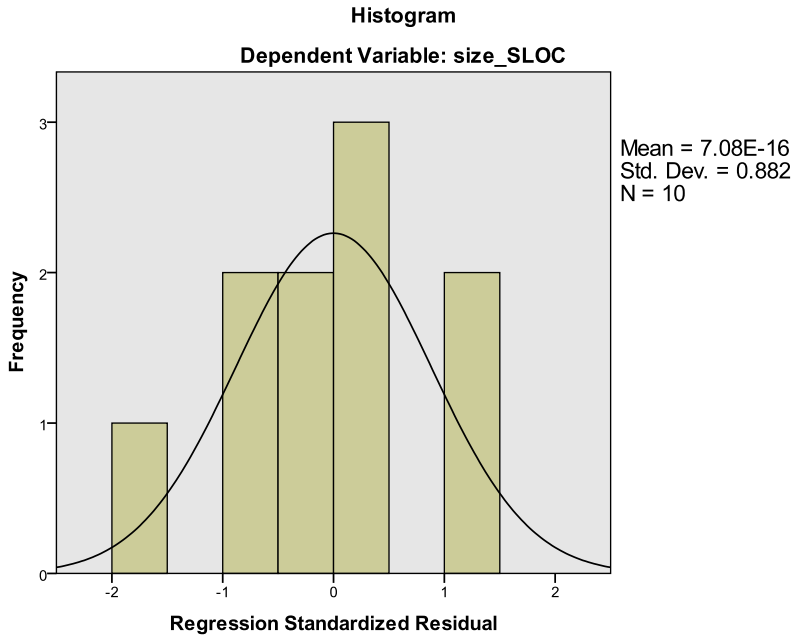

Figure 1. Histogram of standardized residual

5.4 Model Assessment and Validation

We utilized the LOOCV method to evaluate the accuracy of our proposed model, employing various accuracy metrics i.e. MMRE, MdMRE, Pred(25), MAR, SSE, and MSE. The values of these accuracy metrics for our proposed model can be found in Table 8 (refer to Column 3). These findings indicate that our proposed model demonstrates predictive capabilities, as both the MMRE and MdMRE values are below 0.20. Furthermore, the Pred(25) value falls within an acceptable range (greater than 0.75). In accordance with prior research [44, 45], the outcomes of our proposed model are considered good. Therefore, in response to RQ1, we can confidently state that our proposed model achieves a high level of accuracy in estimating software size measured in SLOC.

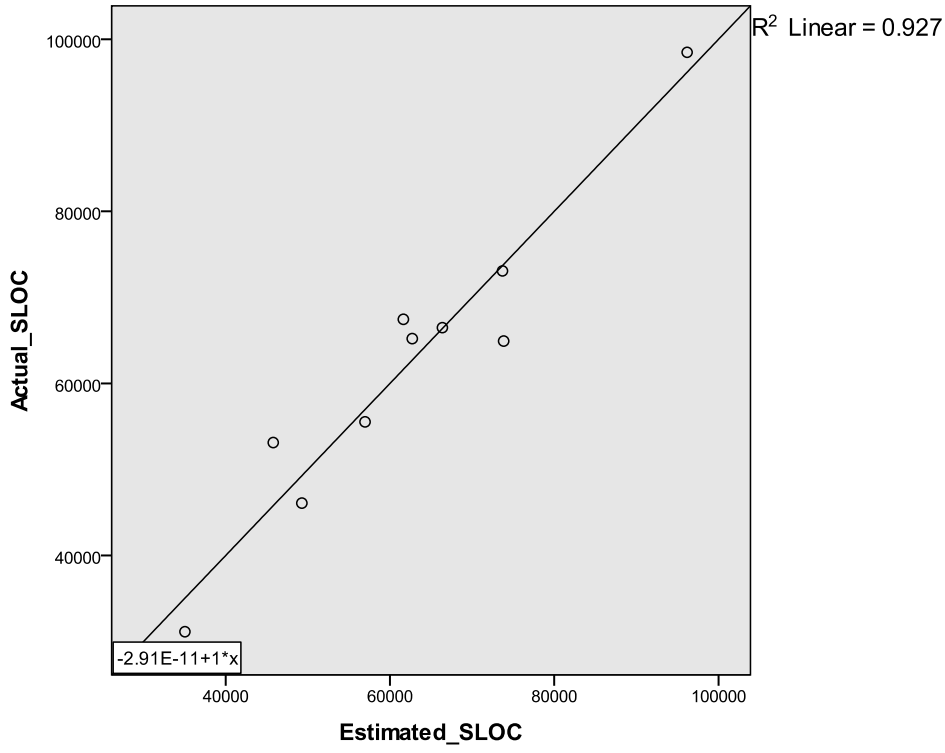

Figure 2. Linear regression line for the proposed model

5.5 Comparison with Existing Models

Table 8 also presents a comparison of the estimation accuracy of the proposed model (Model #1) with the two existing early software size estimation models (Model #2 and Model #3) developed without incorporating security-based NFRs related to data entry validations.

All models perform identically with respect to Pred(25). However, as is evident from the Table 8, the proposed model is significantly superior to the existing two models with respect to all other accuracy measures. It achieves significant improvement in the estimation accuracy compared to Model #2 (i.e. ADAF-adjusted ACD metrics-based model) with respect to MMRE (18 % reduction), MdMRE (18 % reduction), MAR (21 % reduction), SSE (25 % reduction), and MSE (25 % reduction). Since our proposed model attains reduced values for these accuracy metrics, it signifies an improved fit, primarily due to the integration of security-based NFRs associated with data entry validations in the proposed model.

Accuracy Metric	Formula	Proposed Model (Model #1)	Model #2	Model #3
MMRE	$MMRE = \frac{1}{n}\sum_{i=1}^{n} MRE_i$	0.065	0.079	0.086
MdMRE	$MdMRE = MEDIAN_{i=1,n}[MRE_i]$	0.054	0.066	0.074
Pred(25)	$Pred(25) = \frac{1}{n}\sum_{i=1}^{n} \begin{cases} 1, & \text{if } MRE_i \leq \frac{25}{100}, \\ 0, & \text{otherwise,} \end{cases}$	1	1	1
MAR	$MAR = \frac{1}{n}\sum_{i=1}^{n} \lvert y_i - \hat{y}_i \rvert$	3 613	4 576	5 017
SSE	$SSE = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$	206 651 110	273 765 731	365 546 841
MSE	$MSE = \frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2$	20 665 111	27 376 573	36 554 684

Table 8. Comparison summary

Additionally, our proposed model (Model #1) significantly enhances the accuracy of estimation compared to Model #3 (OCP-based model) by reducing MMRE by 24 %, MdMRE by 27 %, MAR by 28 %, SSE by 43 %, and MSE by 43 %.

Since our proposed model outperforms both existing early software size estimation models by achieving a significant reduction in errors, it can be said in response to RQ2 that the incorporation of security-based NFRs related to the data entry validations in the proposed model brings the significant improvement in estimation accuracy vis-à-vis the two existing early software size estimation models.

6 WORKED-OUT EXAMPLE

This section provides guidance to practitioners on utilizing our proposed model for early estimation of SLOC in a new software project, using a practical example.

Let us consider a scenario where the project manager of a software company is assigned the task of overseeing a new software project, specifically the development of a hospital management system (HMS), using the VB.NET programming language from the ground up. To estimate the SLOC for this project, the project manager can employ our proposed model, which was developed based on industrial VB.NET projects (refer to Table 7, Sr. #1).

Initially, the project management team provides the project manager with the initial documentation and the ACD for the HMS. From this ACD, the project manager can easily extract the NOM. Let us assume the NOM for the HMS is 240. The MADAF for NOM, which is 10.2 in this case, is also provided to the project man-

ager. The MADAF is calculated using historical data maintained by the software company, following the steps mentioned in Section 3.2.

Next, the project manager can calculate the ANOM using Equation (3), as demonstrated in Table 9, Step 1. Additionally, by analyzing the documentation of the HMS, the project manager can calculate the SPDEV specific to the HMS, following the steps outlined in Section 3.1. Let us assume the SPDEV for the HMS is determined to be 1271.

Finally, these values, i.e., ANOM = 2 448 and SPDEV = 1 271, can be entered into the proposed early SLOC estimation model (refer to Table 7, Sr. #1). Consequently, the estimated SLOC for this HMS is approximately 61 600, as illustrated in Table 9, Step 2.

Step 1. Collect/Calculate Input Metrics
ADAF-adjusted NOM (ANOM) = NOM $*$ MADAF(NOM)
ANOM = $240 * 10.2$
ANOM = 2 448
SPDEV = 1 271

Step 2. Calculate the Estimated Software Size
SLOC = $-35\,820.07 + 18.93 * \text{ANOM} + 40.21 * \text{SPDEV}$
SLOC = $-35\,820.07 + 18.93 * 2\,448 + 40.21 * 1\,271$
SLOC = 61 628

Table 9. Software size estimation based on the proposed model

7 THREATS TO VALIDITY

We used four criteria (i.e. internal validity, construct validity, external validity, and reliability) provided by Yin [33] to discuss the threats to the validity of this study.

In the case of the dependent variable, the construct validity threat related to the software size (SLOC) was reduced by calculating SLOC automatically using a reliable tool called Understand. Moreover, SPDEV values of projects were first calculated by the first author of this paper and then reviewed and validated by experienced collaborators from the relevant software house. To mitigate the threat related to the internal validity of this study, we assessed the estimation accuracy of the proposed model using a variety of accuracy metrics obtained through the LOOCV method.

Since this study was conducted using a relatively small dataset provided by a single software house, the potential for generalization of the results is limited. However, we used adequate statistical methods to perform this study and obtained promising results which are statistically significant. We believe that these results shed adequate light on the importance of the use of NFRs in software size estimation. Nonetheless, to facilitate other researchers and practitioners, we have also provided a generic repeatable approach which can be applied to any other dataset. Finally,

to improve reliability (which refers to the degree to which the research data and findings are independent of researchers i.e. same results would be obtained when repeating the study [33]), the data and steps of this study were all reviewed by a collaborator.

8 CONCLUSIONS AND FUTURE WORK

Past research has explored the quantification of NFRs in terms of functional size measures such as COSMIC FSM. Some researchers have also looked at the impact of some NFRs such as performance and usability on software size. However, no previous work has focused on exploring the use of security-based NFRs related to data entry validations for estimating the physical size (i.e. SLOC) of a system at an early stage (i.e. project inception). Similarly, most previous work lacks validation of the proposed approaches using a data set comprising real-life industrial projects. Validations reported in the literature use case studies comprising only a few (no more than 3) projects. This research attempts to fill these gaps by evaluating the impact of security-based NFRs on early software size estimation. For this purpose, first security-based NFRs related to data entry validations were utilized to calculate SPDEV for a dataset comprising 11 real-life MIS projects. These SPDEV values were then used along with ADAF-adjusted ACD metrics to build and validate an early SLOC estimation model.

This model was assessed, validated, and compared (concerning the estimation accuracy) with two existing early software size estimation models (i.e. ADAF-adjusted ACD metrics-based model and OCP-based model) built using the same industrial dataset. Results have shown that this model outperforms the existing models and significantly reduces errors. This evidence suggests that including security-based NFRs in an early software size estimation model can lead to a significant improvement in estimation accuracy.

Even though the results of this initial investigation are promising, the dataset currently used in this study is relatively small and is collected from a single software development organization. In the future, a bigger dataset comprising more real-life projects developed in different languages and environments may be utilized to undertake a more thorough investigation of the impact of security-based NFRs on early SLOC estimation. Moreover, the effect of other categories of NFRs on early SLOC estimation may also be investigated empirically.

REFERENCES

[1] Galorath, D. D.—Evans, M. W.: Software Sizing, Estimation, and Risk Management: When Performance Is Measured Performance Improves. Auerbach Publications, 2006, doi: 10.1201/9781420013122.

[2] Boehm, B.—Clark, B.—Horowitz, E.—Westland, C.—Madachy, R.—Selby, R.: Cost Models for Future Life Cycle Processes: COCOMO 2.0. Annals of Software Engineering, Vol. 1, 1995, No. 1, pp. 57–94, doi: 10.1007/BF02249046.

[3] Silva, S. E. F.—Côrtes, M. L.: Use of Non-Functional Requirements in Software Effort Estimation: Systematic Review and Experimental Results. 2017 5th International Conference in Software Engineering Research and Innovation (CONISOFT), 2017, pp. 1–9, doi: 10.1109/CONISOFT.2017.00008.

[4] Ertuğrul, E.—Baytar, Z.—Çatal, Ç.—Muratli, Ö. C.: Performance Tuning for Machine Learning-Based Software Development Effort Prediction Models. Turkish Journal of Electrical Engineering and Computer Sciences, Vol. 27, 2019, No. 2, pp. 1308–1324, doi: 10.3906/elk-1809-129.

[5] 24765, I.: Systems and Software Engineering — Vocabulary. International Organization for Standardization (ISO), 2017.

[6] Symons, C.: Guideline on Non-Functional and Project Requirements. COSMIC, 2015.

[7] Ungan, E.—Trudel, S.—Abran, A.: Analysis of the Gap Between Initial Estimated Size and Final (True) Size of Implemented Software. In: Salmanoglu, M., Coskuncay, A. (Eds.): Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement (IWSM Mensura 2018). CEUR Workshop Proceedings, Vol. 2207, 2018, pp. 123–137, https://ceur-ws.org/Vol-2207/IWSM_Mensura_2018_paper_10.pdf.

[8] 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.

[9] 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.

[10] Ayyildiz, T. E.—Koçyiğit, A.: Size and Effort Estimation Based on Problem Domain Measures for Object-Oriented Software. International Journal of Software Engineering and Knowledge Engineering, Vol. 28, 2018, No. 2, pp. 219–238, doi: 10.1142/S0218194018500079.

[11] Prykhodko, S.—Shutko, I.—Prykhodko, A.: 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.

[12] 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.

[13] Alwan, Z. S.—Younis, M. F.: Detection and Prevention of SQL Injection Attack: A Survey. International Journal of Computer Science and Mobile Computing, Vol. 6, 2017, No. 8, pp. 5–17.

[14] Daud, M.—Malik, A. A.: Improving the Accuracy of Early Software Size Estima-

tion Using Analysis-to-Design Adjustment Factors (ADAFs). IEEE Access, Vol. 9, 2021, pp. 81986–81999, doi: 10.1109/ACCESS.2021.3085752.

[15] IFPUG: Software Non-Functional Assessment Process (SNAP) – Assessment Practices Manual (APM), Release 2.4. 2017.

[16] Kassab, M.—Daneva, M.—Ormandjieva, O.: Scope Management of Non-Functional Requirements. 33rd EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO 2007), IEEE, 2007, pp. 409–417, doi: 10.1109/EUROMICRO.2007.53.

[17] Kassab, M.—Ormandjieva, O.—Daneva, M.—Abran, A.: Non-Functional Requirements Size Measurement Method (NFSM) with COSMIC-FFP. In: Cuadrado-Gallego, J. J., Braungarten, R., Dumke, R. R., Abran, A. (Eds.): Software Process and Product Measurement (Mensura IWSM 2007). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 4895, 2007, pp. 168–182, doi: 10.1007/978-3-540-85553-8_14.

[18] Kassab, M.—Daneva, M.—Ormandjieva, O.: A Meta-Model for the Assessment of Non-Functional Requirement Size. 2008 34th Euromicro Conference Software Engineering and Advanced Applications, IEEE, 2008, pp. 411–418, doi: 10.1109/SEAA.2008.58.

[19] Symons, C.: A Guide to Software Size Measurement, v. 1.0. The COSMIC Group, 2020, https://cosmic-sizing.org/wp-content/uploads/2020/08/Measuring-software-size-v1.0-August-2020.pdf.

[20] Peng, X.—Lee, S. W.—Zhao, W. Y.: Feature-Oriented Nonfunctional Requirement Analysis for Software Product Line. Journal of Computer Science and Technology, Vol. 24, 2009, No. 2, pp. 319–338, doi: 10.1007/s11390-009-9227-2.

[21] Fellir, F.—Nafil, K.—Touahni, R.: Analyzing the Non-Functional Requirements to Improve Accuracy of Software Effort Estimation Through Case Based Reasoning. 2015 10th International Conference on Intelligent Systems: Theories and Applications (SITA), IEEE, 2015, pp. 1–6, doi: 10.1109/SITA.2015.7358402.

[22] Aamodt, A.—Plaza, E.: Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. AI Communications, Vol. 7, 1994, No. 1, pp. 39–59, doi: 10.3233/AIC-1994-7104.

[23] Ochodek, M.—Ozgok, B.: Functional and Non-Functional Size Measurement with IFPUG FPA and SNAP – Case Study. In: Silhavy, R., Senkerik, R., Kominkova Oplatkova, Z., Prokopova, Z., Silhavy, P. (Eds.): Software Engineering in Intelligent Systems. Springer, Cham, Advances in Intelligent Systems and Computing, Vol. 349, 2015, pp. 19–33, doi: 10.1007/978-3-319-18473-9_3.

[24] IFPUG: Function Point Counting Practices Manual, Release 4.3.1. 2010.

[25] Mansoor, H.—Ochodek, M.: Towards Semi-Automatic Size Measurement of User Interfaces in Web Applications with IFPUG SNAP. 2016 Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement (IWSM-MENSURA), IEEE, 2016, pp. 191–194, doi: 10.1109/IWSM-Mensura.2016.037.

[26] Ungan, E.—Trudel, S.—Poulin, L.: Using FSM Patterns to Size Security Non-Functional Requirements with COSMIC. Proceedings of the 27th International

Workshop on Software Measurement and 12th International Conference on Software Process and Product Measurement (IWSM Mensura '17), 2017, pp. 64–76, doi: 10.1145/3143434.3143461.

[27] Trudel, S.—Desharnais, J. M.—Cloutier, J.: Functional Size Measurement Patterns: A Proposed Approach. 2016 Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement (IWSM-MENSURA), IEEE, 2016, pp. 23–34, doi: 10.1109/IWSM-Mensura.2016.016.

[28] Meridji, K.—AlMakhadmeh, K.—Al-Sarayreh, K. T.—Abuljadayel, A.—Khalaf, M. N.: Towards a Requirements Model of System Security Using International Standards. International Journal of Software Engineering and Its Applications, Vol. 9, 2015, No. 4, pp. 139–164.

[29] Meridji, K.—Al-Sarayreh, K. T.—Abran, A.—Trudel, S.: System Security Requirements: A Framework for Early Identification, Specification and Measurement of Related Software Requirements. Computer Standards and Interfaces, Vol. 66, 2019, Art. No. 103346, doi: 10.1016/j.csi.2019.04.005.

[30] Al-Sarayreh, K. T.—Alenezi, M.—Zarour, M.—Meridji, K.: A Reference Measurement Framework of Software Security Product Quality (SPQNFSR). IET Information Security, Vol. 15, 2021, No. 1, pp. 23–37, doi: 10.1049/ise2.12002.

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

[32] Kim, S.—Lively, W. M.—Simmons, D. B.: An Effort Estimation by UML Points in Early Stage of Software Development. In: Arabnia, H. R., Reza, H. (Eds.): Software Engineering Research and Practice (SERP 2006). CSREA Press, 2006, pp. 415–421.

[33] Yin, R. K.: Case Study Research: Design and Methods. 5th Edition. SAGE Publications, 2014.

[34] P@SHA: Pakistan Software Houses Association for IT and ITeS. https://www.pasha.org.pk/.

[35] Mukaka, M. M.: A Guide to Appropriate Use of Correlation Coefficient in Medical Research. Malawi Medical Journal, Vol. 24, 2012, No. 3, pp. 69–71.

[36] Montgomery, D. C.—Peck, E. A.—Vining, G. G.: Introduction to Linear Regression Analysis. 5th Edition. John Wiley and Sons, 2013.

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

[38] Cook, R. D.: Detection of Influential Observation in Linear Regression. Technometrics, Vol. 19, 1977, No. 1, pp. 15–18, doi: 10.1080/00401706.1977.10489493.

[39] 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.

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

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

[42] Hilbe, J. M.—Robinson, A. P.: Methods of Statistical Model Estimation. Chapman and Hall/CRC Press, 2013, doi: 10.1201/b14932.

[43] O'BRIEN, R. M.: A Caution Regarding Rules of Thumb for Variance Inflation Factors. Quality and Quantity, Vol. 41, 2007, No. 5, pp. 673–690, doi: 10.1007/s11135-006-9018-6.

[44] CONTE, S. D.—DUNSMORE, H. E.—SHEN, V. Y.: Software Engineering Metrics and Model. Benjamin-Cummings Publishing Co., Inc., 1986.

[45] HASTINGS, T. E.—SAJEEV, A. S. M.: A Vector Based Approach to Software Size Measurement and Effort Estimation. IEEE Transactions on Software Engineering, Vol. 27, 2002, No. 4, pp. 337–350, doi: 10.1109/32.917523.

Marriam DAUD is currently working as an Assistant Professor at the Department of Software Engineering, National University of Modern Languages (NUML), Islamabad, Pakistan. She received her M.Sc. and 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 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 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, empirical software engineering, software size estimation, and database systems.

Ali Afzal MALIK is currently working as an Assistant 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 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 Institute of Software, Chinese Academy of Sciences (ISCAS). His current research work focuses on areas such as empirical software engineering and software cost estimation.