International Journal of Computer Information Systems and Industrial Management Applications.
ISSN 2150-7988 Volume 12 (2020) pp. 412-423

Received: 23 Jan, 2020; Accepted: 2 June, 2020; Published: 3 December, 2020

Software Enhancement Effort Estimation using Machine Learning Regression Methods

Zaineb Sakhrawi[1], Asma Sellami[2] and Nadia Bouassida[3]

[1] Faculty of Economics and management of Sfax, University of Sfax,
Sened, 2190, Gafsa, Tunisia
zainebsakhraoui40@gmail.com

[2] Higher Institute of Computer Science and Multimedia, University of Sfax,
BP 242. 3021, Sfax, Tunisia
asma.sellami@isims.usf.tn

[3] Higher Institute of Computer Science and Multimedia, University of Sfax,
Cité El Ons 2, N°145, Sfax - Tunisie
nadia.bouassida@isims.usf.tn

***Abstract*: Software enhancement must be carefully planned and taken to satisfy customer change requests, such as adding a new functionality and deleting or changing an existing one. A poorly constructed planning may cause project failures to meet budget targets and deadlines. One of the software project planning activities is effort estimation. In this paper, we investigate the effectiveness and performance of four Machine Learning Regression Methods (MLRM): Ada Boost Regressor (ABR), Gradient Boosting Regressor (GBR), LinearSupport Vector Regression (LinearSVR), and Random Forest Regression (RFR) to predict software requirements enhancement effort. The analysis was based on the results of experiments carried out on real projects in the software industry. These techniques were trained and tested with six software development project datasets including functional requests and the PROMISE repository including enhancement requests. The results of enhancement effort with different machine learning techniques were compared with the enhancement effort obtained from the expert judgement. The best performances were observed with RFR in terms of: MAE (Mean Absolute Error) = 0.040, mean square error (MSE)= 0.045 and root mean square error (RMSE)= 0.215. Therefore, RFR could be recommended for the estimation of software enhancement effort when using expert judgment.**

***Keywords*:** Software Enhancement, COSMIC Functional Change, Software Enhancement Effort estimation, Random Forest Regression (RFR), Linear Support Vector Regression (LinearSVR), Ada Boost Regression (ABReg), Gradient Boosting Regression (GBReg)**.**

I. Introduction

Software enhancement is considered as the critical activity in the software development life cycle. It is defined as "changes made to an existing application where new functionality has been added, or existing functionality has been changed or deleted. This would include adding a module to an existing application, irrespective of whether any of the existing functionality is changed or deleted" [1]. Since changes are frequent throughout the Software Development Life Cycle (SDLC), software project planning should be reviewed frequently. And therefore, software enhancement effort estimation should be accurate.

Considering that effort estimation is one of the main activities of software project planning, it is required to clarify the components of an estimation process. Basically, the quality of the outcomes of an estimation process depends on the quality of its inputs [2]. The size of software to be delivered is recognized as the most significant input variable. As software has increased in size and importance, software development becomes a risky, complex, and costly process [3] In fact, when the software size increases, effort will increase. However, with enhancement, software size may change. Consequently, effort for software enhancement projects needs to be accurately estimated [4]. Although many software estimation models are proposed, expert judgement approach is still used in software industries.

The definition of a requirement change originates from the area of software maintenance and change management [5]. In general, there are two types of change requests: those that are inside the scope and those that are outside the scope of the project. Change requests that are inside the scope involve small corrections to an existing requirement. They usually have minimal impact on the budget or the project progress. On the other hand, change requests that are outside the scope take a considerable amount of time to implement and have a more sizeable impact on the budget [6]. One of the factors that influences the effectiveness of the change acceptance decision is the accuracy of the change effort estimation [7]. Nowadays, an accurate evaluation is a prime goal for risk-free projects [8]. In this context, Machine Learning (ML) techniques which are the most suitable for dealing with modeling of high dimensional

problems are becoming increasingly used to provide efficient solutions [9].

In our previous work [10], we proposed an ontology based-approach for classifying change requests as either Functional changes or Technical changes. Functional changes affect functional requirements, while Technical changes affect nonfunctional requirements as described in the ISO 25010 quality model and Project requirements and constraints. In this paper, we complement this work by measuring the functional size of change request (*i.e.*, enhancement request) using COSMIC FSM method (ISO 19761). The functional size of the change will provide a more realistic evaluation of the change request [11]. Based on the expert judgment approach, we estimate the effort for software enhancement. We propose to assess how well functional change is correlated with enhancement effort estimation and, which of the MLRM provides accuracy for the effort estimation. In order to improve the estimated enhancement effort, we will use four popular MLRM. The main emphasis of this work is to investigate the application of MLRM to achieve superior enhancement effort estimation results by using not only feature selection techniques but also parameter tuning techniques (K-fold-cross validation) on datasets created based on real software development projects and Expert judgement.

The remainder of this paper is organized as follows: section 2 presents the related work on software enhancement effort estimation where MLRM are used. Section 3 presents the MLRM used in this study (Ada Boost Regressor (ABR), Gradient Boosting Regressor (GBR), Linear Support Vector Regression (LinearSVR) and Random Forest Regression (RFR)). Section 4 gives a detailed description of our research method. Section 5 presents the experiments and results where prediction accuracy of four techniques is compared with Expert judgement and addresses the threats to validity. Whereas Section 6 presents the discussion and the limitations of our study. Finally, section 7 provides our conclusions along with a discussion and outlines future work.

II. Related Work

Many researchers proposed to build maintenance enhancement estimation models in pursuit of “accurate” estimates. In this section, we show the results of literature study in which we have investigated different ML techniques with regard to their capabilities to generate accurate estimation models.

There are two current models that have been widely used to estimate change effort which are algorithmic and non-algorithmic models [7]. Estimating maintenance effort using the algorithmic model is based on mathematical formulas. In non-algorithmic models, the estimation can be done by using historic projects and previous experiences. The objective of the maintenance effort estimation is to estimate the amount of work and time required in implementing the particular changes. Expert judgment is one of the non-algorithm traditional techniques that is frequently used in the software cost estimation in the early phases of software project life cycle [12].

Several models have been proposed for software maintenance effort estimation (SMEE). In this study we will investigate the use of four MLRM for enhancement estimation. Accurate estimation is essential to choose for each software application model development [8]. 18 studies proposing SMEE models were published between 1995 and 2020. The models in these 18 studies were statistical regressions [13]–[18], neural networks [18]–[23], SVR [24], rule based [21],[24], Bayesian network [25], analogy [26], and a pattern recognition approach termed optimized set reduction[18], General Regression [23], Support Linear Regression models [23], Support Vector Regression [24] and decision trees stochastic gradient boosting [27].

Table 1 summarizes the three recent research studies related to software maintenance (enhancement) effort estimation.

Results showed that there was not a statistically significant difference in the prediction accuracy among the proposed models. A major challenge for the research community is to develop a good theoretical understanding for maintenance and evolution, which scales to industrial applications [28]. Several studies lack clarity on how the data were prepared and used, which makes it difficult to compare results among studies as well as replicate them [29].

The ISO/IEC 14764:2006 define software maintenance in four types: corrective, adaptive, preventive and perfective [30]. The IEEE 14764 classifies adaptive and perfective maintenance as enhancements and corrective and preventive maintenance as correction. The type of maintenance considered in our work is maintenance enhancement (ME).

The aim of enhancement effort estimation model is to guide manager in their control or avoidance of excessive costs when dealing with requirements change rework effort.

III. Machine Learning Regression Methods

A review of various machine learning techniques used in estimating enhancement effort revealed that the accuracy of estimates can be achievable [31]–[36]. No method is considered better than the others, strengths and weaknesses are often complementary to each other.

References	Techniques	Maintenance type	Evaluation Metrics	Performance
Lopez et al. [23]	Neural Networks (NN), General Regression Neural Network (GRNN) and Radial Basis Function NN (RBFNN)	Enhancement	Absolute Residuals (AR), Mean of Absolute Residuals (MAR)	In comparing the prediction accuracy among the RBFNN, MLP and GRNN, the RBFNN had the highest confidence level.
Gracia et al.[24]	Support Vector Regression (SVR), Decision Tree (DT)	Enhancement	Mean Relative Error (MRE), Absolute Residual (AR)	The polynomial kernel SVR (PK SVR) was statistically better than statistical regression, neural networks, association rules and decision trees, with 95% confidence
Ceron et al. [27]	Statistical regression (SR), NN, SVR, DT and Stochastic Gradient Boosting (SGB)	Enhancement	Absolute Residuals (AR), Mean of Absolute Residuals (MAR)	The SGB had better prediction accuracy than the other five models.

Table 1. Related Work

Several techniques for learning and classification exist in the literature, but it is unclear which one is definitively better than another in our context. For this reason, we used different experiment learning methods to find the most suitable one for our context. In our experiments, we considered the regression methods that are available. The goal of regression methods is to build a function f(x) that adequately maps a set of independent variables (X1, X2,..., Xn) into a dependent variable Y [37]. In our case, we aim to employ regression methods using a training dataset to predict the total enhancements effort for software projects in man-hours. The four MLRM used in our research method are as follow:

- Adaboost Regressor (ABReg),
- Gradient Boosting Regssor (GradientBoostingReg)
- Random Forest Regression (RFR),
- Linear Support Verctor Regression (LinearSVR)

Many algorithms have been used for the purpose of boosting, including Adaptive Boosting (Known as AdaBoost). AdaBoost algorithm was first introduced by Freund & Schapire [38]. This algorithm was a solution to many of the difficulties for earlier boosting algorithms [39]. The idea behind AdaBoost is to construct a strong model by combining multiple weak classifiers into one single strong classifier. A weak classifier is a classifier which performs poorly but better than random guessing [40]. On the other hand, the Gradient Boosting Regression (GBR) is one of the most popular and widely used machine learning models today. The strength of using GBR is to solve almost all objective functions, effective in many cases, flexibility with the choice of loss functions [41]. The RFR is proposed by Breiman, it is an improved classification and regression tree method that gained popularity for its robustness and flexibility in modeling the input-output functional relationship appropriately. The SVR was first introduced by Vapnik in 1995 [42]. To generalize the Support Vector algorithm to regression estimation, an analogue of the margin is constructed in the space of the target values using Vapnik's insensitive loss function [43] Variables in the SVR model structure belong to continuous space.

Our constructed algorithm model contains a collection of 600 enhancement requests {ER1, ER2, ER3...ER600} related to six software development project datasets with their corresponding features ({Functional Process Size, Actual Effort, Change (enhancement) Request, Functional Change Size and Estimated Effort}). The aim is to predict change effort ratios of test samples (new enhancement/change). In other words, the goal is to construct a model that minimizes the prediction error for estimating the future samples.

I. Research Method Overview

This study addressed the Enhancement Effort Estimation problem as a regression task. Our research design includes the following four steps (See Figure 1):

1. Gathering Data,
2. Pre-Processing Data and Normalizing Data,
3. Constructing Prediction Models,
4. Evaluating the performances of models

A. Gathering Data

This step involves the data set collection from two types of database. The first database contains System Requirements (SR) collected from use case diagrams, class diagrams, and project tutorials from previously developed real projects in the software industry. The second database contains Requirements Changes (RC) collected from customers' reviews in PROMISE[1]. These sources provide the system contextual requirements including System purpose, System scope and system overview.

[1] http://promise.site.uottawa.ca/SERepository/datasets-page.html

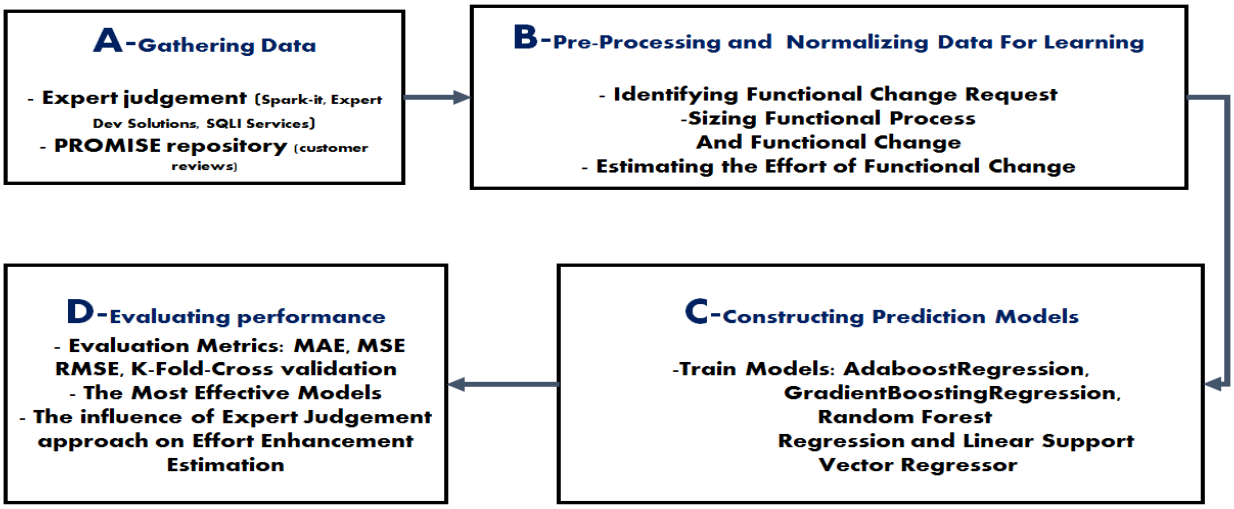


Figure 1. Research method design

Our study takes into account the change request as an input, identifies its types, and measures its function size in terms of CFP units, as well as the actual and the estimated effort of each Functional Process (*i.e.*, functional requirement) derived from experts' opinions.

A. Pre-Processing and Normalizing Data for Learning

1) Identifying Functional Change requests

The collected change requests are pre-processed in our previous work [10]. We used an ontology-based approach to classify change requests as either Functional Change or Technical Change. Technical Changes are further classified into one of the eight ISO 25010 quality characteristics and Project Requirements and Constraints. This ontology-based approach involves three main steps: (i) identification and specification of change requests, (ii) conceptualization of relationship among system requirements change requests, and (iii) an implementation of rules and results generation.

2) Sizing Functional Requirements and Functional Changes requests

Our research method takes into account the functional size of a change request. The ISO/IEC 14143-1 for Functional Size Measurement defines FSM as a means of quantifying the Functional User Requirements (*i.e.*, functions that the functional user has required to be delivered). Since Functional User Requirements can be extracted from the software model before its implementation, COSMIC FSM method can be used for software size measurement and estimation purposes [44]. It is to be noted that none of the effort estimation models used the ISO standard COSMIC [44] which objectively measures the software functional size.

(a) COSMIC FSM: overview

The COSMIC FSM method is an international standardized method for measuring software functionality [45]. The method is designed to be independent of any implementation decisions embedded in the operational artifacts of the software to be measured. COSMIC [36] can be used to approximate the software size at the beginning of the software life-cycle. COSMIC method has been successfully used to size "data manipulation-rich" software, some scientific/engineering software [46]. The COSMIC method measurement unit is the "Cosmic Function Point (CFP)".

Each data movement is measured as 1 CFP. The COSMIC FSM method process [47] includes three steps: Measurement strategy phase, mapping phase and Measurement phase. COSMIC measures the size of a change to software and the size of software that is added, changed or deleted as well. COSMIC functional size measurement is applicable to business, real-time and infrastructure software at any level of decomposition [45]. Our dataset includes four web applications and two business applications.

Over the years, many discussions are centered on software project estimation [48]–[50] And, researchers continue to propose new effort estimation models to achieve effort prediction performance. Typically, these models are classified as either Algorithmic or non-algorithmic models.

Some estimation models depend on the stakeholders experience and skills level and require to spend more time to understand both user requirements and program specifications, or to complete program design [51]–[53]. For many years, expert judgment was the main choice for effort estimation.

(b) Applying COSMIC FSM method

This section introduces the use of COSMIC FSM process as described in [54]. The main parameters that must be identified in Measurement strategy phase are detailed as follows:

- The purpose: Estimating Functional Change Effort.
- Overall scope: Sizing change request and estimating the effort based on Expert Judgment
- Functions users: The functional users in this case are human users.
- Layer: web application, business application.
- Level of granularity: high level of granularity

In order to visualize the interaction between the defined key parameters, we used two different case studies. For web application, the context diagrams are used to show the measured software, along with its functional users, the boundary, the persistent storage and the data movements [55]. For business application, the (human) functional users of the application have no knowledge of how the application is

physically distributed over the PC and the main-frame [56].
A functional user starts a functional process in response to a triggering event. For both web and business application, the triggering events are the users' inputs. Hence, the types of the data movements (Entry (E), Exit (X), Read (R) or Write (W)) were defined.
In Measurement phase, the identified data movements, based on some common word cases [56] such as create, select, delete, add, share and display, were counted. The identified data movements type can be repeatedly executed by the user. This has facilitated the measurement process. In our case we will use the method proposed by Heeringen et al [57]. The size of a functional process is the sum of all its data movements. The Functional Size of a Functional Process (noted by FS(FP)) (*i.e.*, UC) is given by Equation 1.

$$FS(FPi) = \sum FS(Entries) + \sum FS(eXits) + \sum FS(Reads) + \sum FS(Writes) \quad (1)$$

Where:

- FS(FPi): is the functional size of the functional process FPi.

The functional size of a software is obtained by performing an arithmetic addition of the functional sizes of its functional processes (see Equation 2)

$$FS(SW) = \sum FS(FPi) \quad i=1 \quad (2)$$

where:

- FS(SW): the functional size of the software SW.
- FS(FPi): the functional size of the functional process FPi.
- n: the number of functional processes in the software.

Our dataset contains six software development projects, including 600 FP. After identifying all the data movements for each FP, Table 2 illustrates the application of COSMIC for an example of three FP related to the "SOCOMENIN" project that has a functional size of 142 CFP. And an example of three FP related to the "Mobile game" project having a total functional size of 175 CFP.

3) Estimating the Enhancement effort based on Expert judgment approach

When data is gathered, there are some limitations to a number of scenarios. In these situations, the Expert Judgement approach is 'good' to be used [58]. It provides fast estimation for unique or new projects [59], [60]. It is useful when an organization does not have any historical data in database [61]. It provides estimates which are adjusted and calibrated to the past of organization by means of expert experience. It does not require any historical data.

Years of software Experience	Numbers of estimators
3 years	2
4 years	3
10 years	2

Table 3. Expert Judgement Experience.

In our studies, as shown in Table 3, we have asked seven estimators having at least three years of experience. Each Functional Process and its corresponding actual effort derived from the Expert judgments is provided in Table 2. The results are based on a set of estimators' information such as, design requirements, available resources, base product/source code, available software tools, and quality requirements.

Project	Functional Process	Status	CFP	Actual Effort
Mobile game application	Login	Completed	E=3; X=3; R=1; W=1; total=8CFP &	2h/per
	send localization information	Completed	E=3; X=3; R=1; W=0; total= 7CFP	4h/per
	store information	Completed	E=2 ;X=2; R=1; W=0; 5CFP	2h/per
SOCOMENIN	Add Customer	Completed	E =1X =1;R =1;W=1; total= 4CFP	1h/per
	Add invoice	Completed	E=1; X=1 ;R=1 ;W=1; total= 4CFP	3h/per
	Send events notification	Completed	E=2; X=2; R=1; W=0; total= 5CFP	2h/per

Table 2. Change Request size based on COSMIC CFP.

Project	Functional Process	Functional Change	Change Type	CFP	Actual Effort
Mobile Game application	Login	Personalize Information &	Modify	E=3; X=3; R=1; W=1; total= 8CFP	3h/per
	send localization information	Add notification for localization	Add	E=3; X=3; R=1; W=0; total =7CFP	7h/per
SOCOMENIN	Add Customer	Personalize Information	Modify	E=3; X=3; R=1; W=1; total= 8CFP	3h/per
	Add invoice	Personalize Information for invoice	Add	E =3; X =3; R =1; W=1; total= 8CFP	3h/per

Table 4. Enhancement Effort Estimation based on Expert Judgement

Table 4 presents the change requests that are gathered from our Ontology-based approach. A number of FP are affected by functional change request. Using COSMIC method, the functional size of each functional process as well as the functional size of the change are provided. The measurement results are obtained from the sum of all FP sizes. Each estimated effort to handle a change request is based on Expert judgments. Thus, estimating the effort of change request (*i.e.*, estimating the enhancement effort) is not an easy task for both the manager and the development team when software is being developed. In this situation, the results are based on a set of estimators' information such as, previous history of the product (e.g. previous changes), Functional Size of the new function, similar previous implementations, amount of new code, deadline pressure and expected life-time of the product.

A. Constructing Prediction Model

The selected MLRM are trained and tested for various experiments using features. We used the Google Collaboratory python programming language to develop our prediction model. Google Collaboratory is widely known as Google Colab is an open source service provided by Google to any person having a Gmail account. Google Colab[2] provides GPU for research to the people who do not have enough resources or cannot afford one. This section conducted series of experiments to investigate the application of COSMIC method and Expert Judgements. Our proposed prediction method was evaluated through six software development project datasets.

B. Evaluating Performance

Many researchers used different metrics to evaluate the performance of their proposed model [62]–[66] In our Context, the outcomes of our constructed enhancement effort estimation models are compared with the widely used set of evaluation metrics such as mean square error (MSE), root mean square error (RMSE) and mean absolute error (MAE). In this section, two types of experiments are conducted. In the first set of experiments, dataset is randomly split into two subsets, a training set and a test set. And, the second set of experiments is conducted using popular tenfold cross validation method.

[2] https://colab.research.google.com/notebooks/welcome.ipynb

1) Error metrics

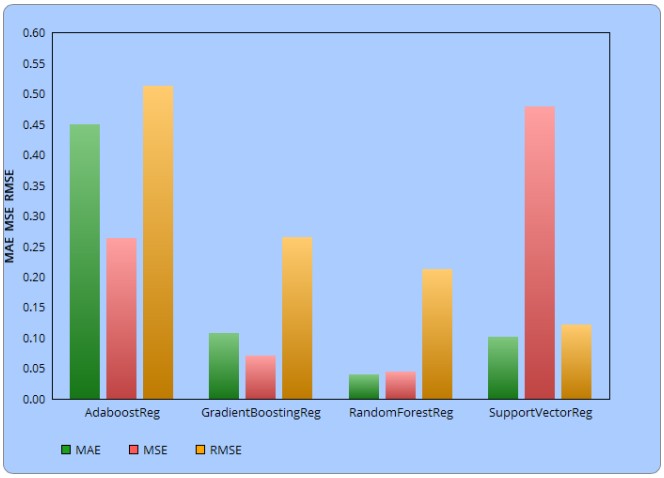
The predicted values are compared with actual target values and prediction errors are computed in the form of MAEs, MSEs and RMSEs as demonstrated in Table 5.

Method/parameters	MAE	MSE	RMSE
ABReg	0.450	0.263	0.513
GBReg	0.108	0.070	0.265
RFR	0.040	0.045	0.215
LinearSVR	0.100	0.479	0.190

Table 5. Prediction analysis using MAE, MSE and RMSE.

All error measurements indicate quiet values. It is evident from the results (see Figure 2) that RFReg method delivers the best performance when compared with the other three

Figure 2. Prediction analysis using MAE, MSE and RMSE



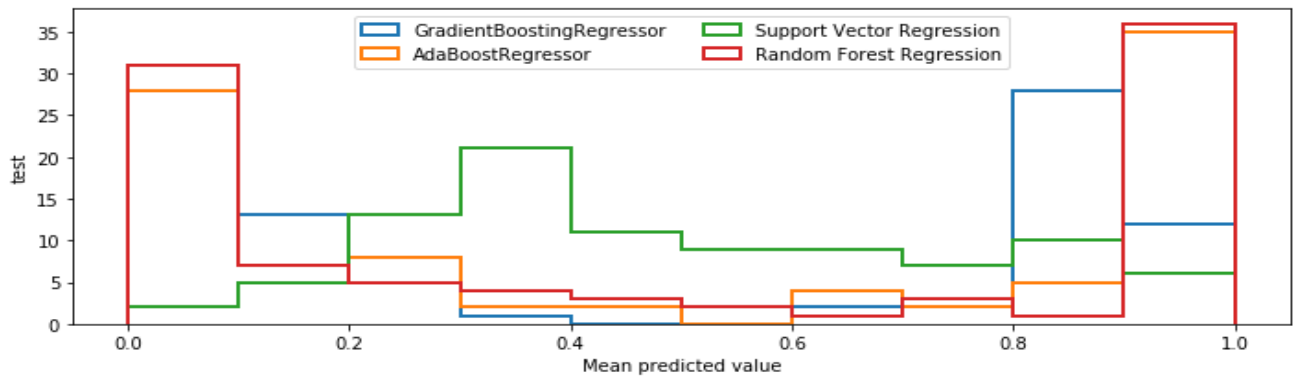
MLRM. It shows the evidence of its powerful predictive capacity. In addition, the GBReg presents good results. However, the bad results are presented by the ABReg method.

2) Simple split

For the first set of experiments, the classic approach is to do a simple 80%-20% split, sometimes with different values like 70%-30% or 90%-10%. Using classic method, we often split

Figure 3. Mean Predicted value

our data into training and validation/test sets. The training set



is used to train the model, and the validation/test set is used to validate data that it has never seen before. Results of the experiments (see Figure 3) revealed that Random Forest provides better predictive performance compared other models.

1) Cross validation

In cross-validation, we do more than one split. We can do 3, 5, 10 or any K number of splits. These splits are called Folds, and there are many strategies that can be used to create these folds. In our model, we used 10-fold for cross validation. It is adapted in all sorts of experiments to analyze the performance of four MLRM. By using Cross-Validation, we are able to get more metrics and draw important conclusions both about our algorithm and our data. One of the most obvious reasons to do cross validation is that it helps us better use our data, and it gives us much more information about our algorithm performance. We also used the r2 (coefficient of determination) regression score function for cross validation (Algorithm 1). Table 6 illustrates the results of using these two metrics (accuracy/prediction).

Techniques	ABReg	GBReg	RFR	LSVR
Accuracy_lf	0.8203	0.8634	0.8836	0.7168
Cross-Predicted (K-Fold)	0.8066	0.8496	0.8175	0.8011

Table 6. 10-Fold Cross Validation accuracy.

2) Features Importance

We have conducted a set of experiments in order to investigate the influence of each feature on the estimated effort based on the best model results. Features including {Functional Process Size, Actual Effort, Change (enhancement) Request, Change (enhancement) Functional Size, Estimated Effort} are shown in Figure 4. Results demonstrate that features {Actual Effort, Functional Change (enhancement) and Estimated Effort} have a good influence on the delivered results. This clearly shows the utility of Expert judgments in predicting enhancement effort. Figure 4 presents the results.

In order to obtain more performance prediction, we used only the following features {Actual effort and Change

Figure 4. Importance of Features

(enhancement) request}. Results of the experiments (see Figure 5) revealed that RFReg and GtBReg deliver better mean score cross validation prediction when compared with other models. However, the bad results are delivered by Linear SVR.

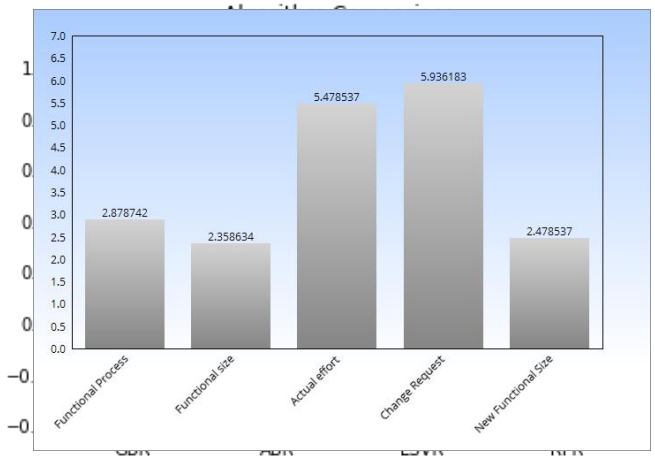


Figure5. Mean Predicted value

Algorithm 1: 10-Cross-Validation with r2_score

```
#Predicting the enhancement effort using test set
y_pred_rfr= RFR.predict(X_test)
#Random Forest Regression Accuracy with test set
accuracy_rfr = metrics.r2_score(y_test, y_pred_rfr)
print("RFR Accuracy:," accuracy_rfr)
#Predicting the enhancement effort using cross validation (KFold method)
y_pred_kf_rfr = cross_val_predict(RFR, X, Y, cv=10 )
#Random Forest Regression Accuracy with KFold method)
accuracy_rfr = metrics.r2_score(Y, y_pred_KFold_rfr)
print(accuracy_rfr)
```

II. Threads to validity

Threats to the validity of our study are pertinent to internal validity, external validity, and finally construct validity.

A. Internal validity

The threat to internal validity in our research is related to the classification of change request as Functional Change (FC) or Technical change (TC), as well as the change Functional Size (FS(FC)). The classification of change request is based on the use of Ontology. The strength of using Ontology is to move from simple classification to semantic classification [66]. As it is described in section 3, using ontology for classification is for improving the quality of data (models input). In addition, our

classification ontology-based approach requires that the request for changes must be defined clearly and completely. Thus, using COSMIC FSM for sizing FC can be used throughout the software life-cycle phases. We identified the FS(FC) at a high level of abstraction of the software life-cycle.

B. *External validity*

The threats to external validity primarily include the degree to which the proposed research method has exactly captured the estimate (*i.e.*, enhancements effort) for software projects in practice. For data quality, we only extracted a small portion of data samples from the used data sets based on Expert Judgment. Figure 6 shows the observed and estimated enhancement effort using RFReg.

```
# the inputs
Xnew = [[ 2 , 142 ,  8 ,  2 ,143 ,  2]]
# make a prediction
ynew = model.predict(Xnew)
# show the inputs and predicted outputs
print(("input:", Xnew[0], "Predicted:", ynew[0]))
# compare the Xnew with the results
print("True value:",[2],"Predicted:",ynew )
print((X[0].reshape(1, -1), Y[0].reshape(1, -1)))
ynew = model.predict(X[0].reshape(1, -1))
print(round(ynew[0]))
```

```
('input:', [2, 142, 8, 2, 143, 2], 'Predicted:', 2.0)
True value: [2] Predicted: [2.]
(array([[ 2,  1,  8,  2, 13,  8]]), array([[3]]))
3.0
```

Figure 6. Mean Predicted value

Figure 7. Mean Predicted value

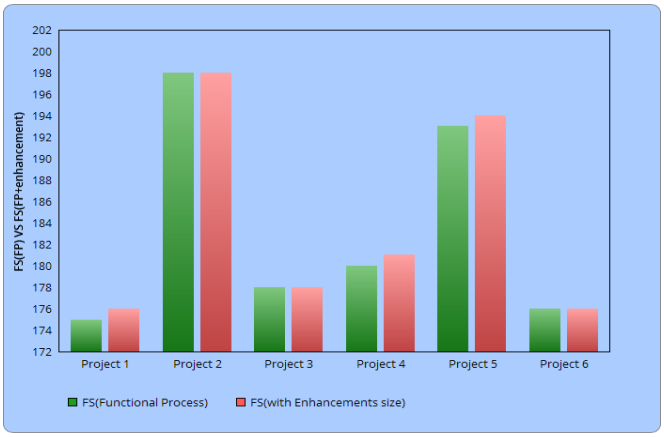
Figure 7 shows the bar graph representation of actual effort values and estimated enhancement effort values with our proposed research method. The bar graph shows three cases.

The first case of Project 1 shows that the estimated enhancement effort can be very close to the actual effort.

The second case of Project 3 shows that the actual effort can

Figure 8. Mean Predicted value

be higher than the estimated enhancement effort. And, the third case of projects 6 shows that the estimated enhancement effort can be higher than the actual effort. Regarding the functional size measurement, Figure 8 shows the bar graph representation of Functional Size of the total Functional processes and the Functional Size of the total enhancement Functional processes in terms of CFP units. The bar graph shows that values are very close. This is due to the fact that our research method takes



into account Functional Process as well as the related enhancement, Process with high level of granularity.

A. *Threats of construct*

The threats of construct validity are related to the relation between theory and observation. In fact, the estimation of enhancement effort in our study is provided using Expert judgment method based on an ontology classification. Moreover, the judgment of enhancement request effort depends on many factors such as enterprise employee experience, categories, etc. The use of the COSMIC method for quantifying the functional requirements and enhancement will be helpful for the development team.

Results (Figure 6,7 and 8) confirm the validity of MLRM as an alternative to traditional estimation methods such as Expert Judgement. This estimation will help experts making decisions whether to accept, defer or deny the enhancement request.

III. Discussion

All the participants in software project recognize the importance of developing an accurate enhancement effort estimate, since it plays a key role in the success of the software project planning. The main idea of our research is to present an effective model for software enhancement effort estimation.

We focused on the importance of semantic classification when using an Ontology-based approach, and therefore we investigate their impacts for a “good” prediction.

The proposed enhancement effort prediction model is quite effective and demonstrates the minimum MAE of 0.040 using a real dataset project as presented in Table 5. After learning, the MLRM were able to produce reasonably accurate predictions. This study and experiment were done to evaluate four machine learning methods Ada Boost Regression (ABR), Gradient Boosting Regression (GBR), Linear Support Vector Regression (LinearSVR) and Random Forest Regression (RFR). These methods are used to predict effort for a new change request that occurs in the software development project. The Random Forest method is established to be the more effective algorithm when compared with the other three methods.

We used two methods for evaluation. The first method used a simple split. The second method used 10-fold cross validation. In addition, we used the R2 score to cross validation. Based on the obtained results, we noted a small value of MSEs, MREs and RMSEs when applying a simple method alone. It demonstrates the effectiveness of the used methods. A good accuracy of 90% is obtained when the 10-fold cross validation with R2 score in the best scenario is used. To identify the effective determinants of the software enhancement effort estimation, we calculate the importance of each feature. Furthermore, a model that uses {Actual effort and change request} features delivers superior performance as compared to a model that uses all proposed features. In addition, a model that incorporates combination of 10-fold cross validation and R2 score demonstrates better performance when compared with a model that uses a simple split (train/test).

Thus, we can conclude that RFReg and GBReg technique improves the accuracy of the estimates.

For future work, other machine learning techniques can be used for prediction problems. In addition, other datasets can be used for experiments and training the model. In this way, more accurate estimations and predictions can be generated. And, more than two techniques can be combined to get more accuracy reaching 100%.

Similar to all empirical studies, this study has some limitations. When the dataset is tested with Deep Learning, we do not achieve good accuracy (0.6). This is due to the lack of data, and that, Deep Learning requires a big number of datasets. So, we aim to extend our database for future work.

IV. Conclusion

All the participants in software project recognize the importance of developing an accurate enhancement effort estimate, since it plays a key role in the success of the software project planning. The main idea of our research is to present an effective model for software enhancement effort estimation.

Enhancement requests effort estimation became a challenging task in the software project planning. We investigated the problem of accurately estimate effort for software enhancement projects. First, we determined the functional size of the requirements enhancement in terms of CFP unit. Then, for each requirements enhancement, we identify its corresponding estimation effort based on the expert judgement approach.

Thereafter, we evaluated 600 enhancement requests using four MLRM: such as Ada Boost Regressor (ABR), Gradient Boosting Regressor (GBR), Linear Support Vector Regression (LinearSVR), and Random Forest Regression (RFR). In addition, the importance of each feature is also examined. The constructed models are tested using datasets of real software projects based on Expert Judgement, COSMIC FSM method and PROMISE repository for enhancement request. The four MLRM have successfully estimated the enhancements effort for the project's dataset. It was found that Random Forest Regression enhancement effort estimation model is more accurate with small MSEs, MREs and RMSEs values results and with quite good performance.

Several extensions can be made. This work will be extended by exploring other forms of ML techniques to estimate enhancement effort for software engineering projects. We aim to extend our dataset to be used for Deep Learning method. The experiment outputs suggest that the suggested Deep Learning method can provide better results and accurately forecast the enhancement request effort. We aim to investigate the use of the COSMIC FSM method at a detail level of granularity as an input of enhancement effort estimation model.

Acknowledgment

The authors would like to thank Mr.Taher Labidi (Primatec Engineering), Mr.Mounir Ktata (EDS(Expert Dev Solutions)), Mr.Hamza Elkar (Spark-it), Ms.Mariem Mellef (SQLI Services) for their support, advice, and expertise about the software development process.

References

[1] International Software Benchmarking Standards Group development and enhancement, Journal of Software: Evolution and Process, version 5.1, pp. 11, 2018.

[2] Abran, Alain. Software project estimation: the fundamentals for providing high quality information to decision makers, Information and Software Technology, 2015.

[3] Tesch, Debbie and Kloppenborg, Timothy J and Frolick, Mark N IT project risk factors: the project management professionals perspective, Journal of computer information systems, version 5.17, pp. 61–69, 22007.

[4] Garcia-Floriano, Andres and Lopez-Martrtin, Cuauhtemoc and Yanez-M arquez, Cornelio and ´Abran, Alain. Support vector regression for predicting software enhancement effort, Information and Software Technology,97, pp. 99–109, 2018.

[5] Mcgee, Sharon and Greer, Des. Sources of Software Requirements Change from the Perspectives of Development and Maintenance, Journal of computer information systems, version 5.17, 2010.

[6] Lederer, Albert L and Prasad, Jayesh. Perceptual congruence and information systems cost estimating, Journal of computer information systems,pp. 50–59 , 1995

[7] Basri, Sufyan and Kama, Nazri and Sarkan, Haslina Md and Adli, Saiful and Haneem, Faizura. An algorithmicbased change effort estimation model for software development, 2016 23rd Asia-Pacific Software Engineering Conference (APSEC), pp. 177–184, 2016.

[8] Singh, AJ and Kumar, Mukesh. Comparative Analysis on Prediction of Software Effort Estimation Using Machine Learning Techniques, Available at SSRN 3565822, 2020.

[9] Susto, Gian Antonio and Schirru, Andrea and Pampuri, Simone and McLoone, Sean and Beghi, Alessandro. ´ Machine learning for predictive maintenance: A multiple classifier approach, IEEE Transactions on Industrial Informatics,11, pp. 812–820, 2014.

[10] Sakhrawi, Zaineb, Sellami, Asma, Bouassida,Nadia. Requirements Change Requests Classification: An Ontology-Based Approach, The International Conference on Intelligent Systems Design and Applications (ISDA), pp.487–496, 2019.

[11] Haoues, Mariem and Sellami, Asma and Ben-Abdallah, Hanene. Functional change impact analysis in use cases: An approach based on COSMIC functional size measurement, Science of Computer Programming, 135, pp. 88–104, 2017.

[12] Molokken, Kjetil and Jorgensen, Magne. Software function, source lines of code, and development effort prediction: a software science validation, 2003 International Symposium on Empirical Software Engineering, 2003. ISESE 2003. Proceedings, pp. 223–230, 2003.

[13] Yu, Liguo. Indirectly predicting the maintenance effort of open-source software, Journal of Software Maintenance and Evolution: Research and Practice,18, pp. 311–332, 2006

[14] Shukla, Ruchi and Misra, AK. Software maintenance effort estimation-neural network vs regression modeling approach, International Journal of Computer Applications,975, pp. 157–169, 2010.

[15] De Lucia, Andrea and Pompella, Eugenio and Stefanucci, Silvio. Assessing effort estimation models for corrective maintenance through empirical studies, Information and Software Technology,15, pp. 3–15, 2005.

[16] Ahn, Yunsik and Suh, Jungseok and Kim, Seungryeol and Kim, Hyunsoo. The software maintenance project effort estimation model based on function points, Journal of Software maintenance and evolution: Research and practice,15, pp. 71–85, 2003.

[17] Kitchenham, Barbara and Pfleeger, Shari Lawrence and McColl, Beth and Eagan, Suzanne. An empirical study of maintenance and development estimation accuracy, Journal of systems and software,64, pp. 57–77, 2002.

[18] Jorgensen, Magne. Experience with the accuracy of software maintenance task effort prediction models, IEEE Transactions on software engineering,21, pp. 674–681, 1995.

[19] Lopez-Martrtin, Cuauhtemoc. Predictive accuracy com- ´ parison between neural networks and statistical regression for development effort of software projects, Applied Soft Computing,27, pp. 434–449, 2015.

[20] Ku, Yan and Du, Jing and Yang, Ye and Wang, Qing. Estimating software maintenance effort from use cases: An industrial case study, 2011 27th IEEE International Conference on Software Maintenance (ICSM),15, pp. 482–491, 2011.

[21] Shukla, Ruchi and Shukla, Mukul and Misra, Arun Kumar and Marwala, Tshilidzi and Clarke, WA. Dynamic software maintenance effort estimation modeling using neural network, rule engine and multi-regression approach, International Conference on Computational Science and Its Applications,15, pp. 157–169, 2012.

[22] Shukla, Ruchi and Misra, Arun Kumar. Estimating software maintenance effort: a neural network approach, Proceedings of the 1st India software engineering conference, pp. 107–112, 2008.

[23] Lopez-Martin, Cuauhtemoc. Predictive accuracy com- ´ parison between neural networks and statistical regression for development effort of software projects, Applied Soft Computing, 27, pp. 434–449, 2015.

[24] Garcia-Floriano, Andres and Lopez-Martin, Cuauhtemoc and Yanez-Marquez, Cornelio and Abran, Alain. Support vector regression for predicting software enhancement effort, Information and Software Technology,97, pp. 99–109, 2018.

[25] Song, Tae-Hoon and Yoon, Kyung-A and Bae, DooHwan. An approach to probabilistic effort estimation for military avionics software maintenance by considering structural characteristics, 14th Asia-Pacific Software Engineering Conference (APSEC'07), pp. 406– 413, 2007.

[26] Leung, Hareton KN, Emilia, Vrijendra. Estimating maintenance effort by analogy, Empirical Software Engineering,7, pp. 157–175, 2002.

[27] Ceron-Figueroa, Sergio and Lopez-Martin, Cuauhtemoc and Yanez-Marquez, Cornelio. Stochastic gradient boosting for predicting the maintenance effort of software-intensive systems, IET Software,pp. 99–109 , 2019.

[28] Bennett, Keith H and Rajlich, Vaclav T. Software main- ´ tenance and evolution: a roadmap, Proceedings of the Conference on the Future of Software Engineering, pp. 73–87, 2000.

[29] Gonzalez-Ladron-de-Guevara, Fernando and Fernandez Diego, Marta and Lokan, Chris. The usage of ISBSG data fields in software effort estimation: A systematic mapping study, Journal of Systems and Software,113, pp.188–215, 2016.

[30] P. Bourque, R. Fairley. Guide to the Software Engineering Body of Knowledge, SWEBOK V3.0, IEEE Computer Society, 2014.

[31] Basgalupp, Marcio P and Barros, Rodrigo C and ´Ruiz, Duncan D. Predicting software maintenance effort through evolutionary-based decision trees, Proceedings of the 27th Annual ACM Symposium on Applied Computing, pp. 1209–1214, 2012.

[32] Kaur, Arvinder and Kaur, Kamaldeep and Malhotra, Ruchika. Soft computing approaches for prediction of software maintenance effort, International Journal of Computer Applications,16, pp. 69–75, 2010.

[33] Dubey, Sanjay Kumar and Rana, Ajay. A fuzzy approach for evaluation of maintainability of object oriented software system, International Journal of Computer Applications,21, pp. 60–73, 2012.

[34] Sharawat, Mr Sandeep. Software maintainability prediction using neural networks, Journal of Environment,5, pp. 750–755, 2012.

[35] Sharawat, Mr Sandeep. Software maintainability prediction using neural networks, Journal of Environment,5, pp. 750–755, 2012.

[36] Mishra, Swati and Sharma, Arun. Maintainability prediction of object oriented software by using adaptive network based fuzzy system technique, International Journal of Computer Applications,9, 2015.

[37] SBraga, Petronio L and Oliveira, Adriano LI and Meira, Silvio RL. Software effort estimation using machine learning techniques with robust confidence intervals, 7th international conference on hybrid intelligent systems (HIS 2007),5, pp. 352–357, 2007.

[38] Freund, Yoav and Schapire, Robert E. A desiciontheoretic generalization of on-line learning and an application to boosting, European conference on computational learning theory,5, pp. 23–37, 1995.

[39] Freund, Yoav and Schapire, Robert and Abe, Naoki. A short introduction to boosting, Journal-Japanese Society For Artificial Intelligence,14, pp. 771-780, 1999.

[40] Hidmi, Omar and Sakar, Betul Erdogdu, Robert and Abe, Naoki.Software development effort estimation using ensemble machine learning, Int J Comput Commun Instrum Eng,4, pp. 143–147, 2017.

[41] Ertugrul, Egemen and Baytar, Zakir and Catal, Cagatay and Muratli, Omer Can. Performance tuning for machine learning-based software development effort prediction models, Turkish Journal of Electrical Engineering & Computer Sciences,27, pp. 1308–1324, 2019.
[42] Breiman, Leo. Random forests, Machine learning,45, pp. 5–32, 2019.
[43] Cortes, Corinna and Vapnik, Vladimir. Support-vector networks, Machine learning,20, pp.273–297, 1995.
[44] Gencel, Cigdem. How to use cosmic functional size in effort estimation models?, Software Process and Product Measurement,284, pp.196–207, 2008.
[45] ISO, TCJTC. Software Engineering-COSMIC-FFP Functional size Measurement Method, ISO,284, pp.196–207, 2011.
[46] Ebert, Christof and Soubra, Hassan. Functional size estimation technologies for software maintenance, 2014 Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement,31, pp.24–29,2014.
[47] Symons, CR and Lesterhuis, A. Introduction to the COSMIC method of measuring software, COSMIC,31, pp.24–29,2014.
[48] Trendowicz, Adam and Jeffery, Ross. Software project effort estimation, Foundations and Best Practice Guidelines for Success, Constructive Cost Model–COCOMO pags,12, pp.277–293,2014.
[49] Trendowicz, Saeed, Ayesha and Butt, Wasi Haider and Kazmi, Farwa and Arif, Madeha. Survey of Software Development Effort Estimation Techniques, Proceedings of the 2018 7th International Conference on Software and Computer Applications,12, pp.82–86,2018.
[50] Nerkar, LR and Yawalkar, PM. Software Cost Estimation using Algorithmic Model and Non-Algorithmic Model a Review, Int J Comput App,2, pp.4–7,2014.
[51] Tawfik, Sherif M and Abd-Elghany, Marwa M and Green, Stewart. A software cost estimation model based on quality characteristics, IWSM (International Workshop in Software Measurement) and MENSURA (International Conference on Software Process and Product Measurement),2, pp.13–31,2007.
[52] Ochodek, Mirosław. Functional size approximation based on use-case names, Information and Software Technology,80, pp.73–88,2016.
[53] Zhang, Cheng and Tong, Shensi and Mo, Wenkai and Zhou, Yang and Xia, Yong and Shen, Beijun. ESSE: an early software size estimation method based on autoextracted requirements features, Proceedings of the 8th Asia-Pacific Symposium on Internetware,80, pp.112– 115,2016.
[54] M. v5.0.1. Common Software Measurement International Consortium, Version 5.0, 2020.
[55] M. v4.0.1. Common Software Measurement International Consortium (COSMIC), Proceedings of the 8th Asia-Pacific Symposium on Internetware,7,2015.
[56] Abran, A and others. Guideline for sizing business applications software, v 1.1, COSMIC, May,2008.
[57] D'Avanzo, Loris and Ferrucci, Filomena and Gravino, Carmine and Salza, Pasquale. Cosmic functional measurement of mobile applications and code size estimation, Proceedings of the 30th Annual ACM Symposium on Applied Computing,16, pp.1631–1636,2015.
[58] Van Heeringen, Harold and Van Gorp, Edwin. Measure the functional size of a mobile app: Using the cosmic functional size measurement method, 2014 Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement,16, pp.11–16,2014.
[59] Suri, PK and Ranjan, Pallavi. Comparative analysis of software effort estimation techniques, International Journal of Computer Applications,48, pp.12–19,2012.
[60] Shekhar, Shivangi and Kumar, Umesh. Review of various software cost estimation techniques, International Journal of Computer Applications,141, pp.31–34,2016.
[61] Bajwa, Sohaib-Shahid. Investigating the nature of relationship between software size and development effort, International Journal of Computer Applications,2008.
[62] Galorath, Daniel D and Evans, Michael W. Software sizing, estimation, and risk management: when performance is measured performance improves, International Journal of Computer Applications,2006.
[63] Idri, Ali and azzahra Amazal, Fatima and Abran, Alain. Analogy-based software development effort estimation: A systematic mapping and review, Information and Software Technology,58, pp. 206–230, 2015.
[64] Bibi, Stamatia and Stamelos, Ioannis. Selecting the appropriate machine learning techniques for the prediction of software development cost, IFIP International Conference on Artificial Intelligence Applications and Innovations,5, pp. 533–540, 2006.
[65] Huang, Sun-Jen and Chiu, Nan-Hsing. Applying fuzzy neural network to estimate software development effort, Applied Intelligence,30, pp. 73–83, 2009.
[66] Labidi, Taher., Sakhrawi, Zaineb., Sellami, Asma., Mtibaa, Achraf. An Ontology-Based Approach for Preventing Incompatibility Problems of Quality Requirements During Cloud SLA Establishment. In International Conference on Computational Collective Intelligence, pp. 663-675,2019.

Author Biographies

Zaineb Sakhrawi received her Master Thesis degree in Computer Science from Higher Institute of Computer Science and Multimedia, Sfax University, Tunisia in 2018. She is currently a PhD student at Sfax University, Tunisia and a member of Multimedia, InfoRmation systems and Advanced Computing Laboratory (MIRACL). Her research interests include software engineering, ontology, cloud computing and machine Learning techniques.

Asma Sellami is teaching at the University of Sfax in Tunisia. Her current research interest includes broadly measurement in Software Engineering, software quality and software project management. She is also working on ISO standards for measuring the functional size of software, and has been involved in developing case study of ISO 19761 (COSMIC FSM Method). She published more than 40 referred conferences, journals, and technical reports. She is currently member of COSMIC Advisory council in Tunisia.

Nadia Bouassida received a Phd in Computer and Information Science from the University of Science of Tunis, Tunisia. Currently, she is Associate Professor at the Department of Computer Science of the Higher Institute of Computer Science and Multimedia at the University of Sfax, Tunisia. She is a member of the Multimedia, Information systems and Advanced Computing Laboratory, University of Sfax. Her research interests include reuse techniques, such as design patterns, Frameworks and Software Product Lines.