Software Maintainability Prediction Model for Object-Oriented Software Systems Based On Sensitivity - Based Linear Learning Method

Sunday Olusanya Olatunji

Abstract

This paper presented a new maintainability prediction model for an object-oriented (OO) software system based on the recently introduced learning algorithm called Sensitivity Based Linear Learning Method (SBLLM) for two-layer feedforward neural networks.. As the number of object-oriented software systems increases, it becomes more important for organizations to maintain those systems effectively. However, currently only a small number of maintainability prediction models are available for object oriented systems. In this work, we develop Sensitivity Based Linear Learning maintainability prediction model for an object-oriented software system. The model was constructed using popular object-oriented metric datasets, collected from different object-oriented systems. Prediction accuracy of the model was evaluated and compared with commonly used regression-based models and also with Bayesian network based model which was earlier developed using the same datasets. Empirical results from simulation show that our SBLLM based model produced promising results in term of prediction accuracy measures authorized in OO software maintainability literatures, better than most of the other earlier implemented models on the same datasets.

College of Computer Science and Engineering, King Fahd University of Petroleum & Minerals, P. O. Box 8283, Dhahran 31261, Saudi Arabia. Email : olatunji@kfupm.edu.sa, s.o.olatunji@gmail.com

Keywords : Sensitivity based linear learning method (SBLLM), Object oriented Software systems, Software Metrics, Software Maintainability prediction models.

1. Introduction

Software maintainability is the process of modification of a software product after delivery to correct faults, to improve performance or other attributes, or to adapt the product to a changed environment. Maintaining and enhancing the reliability of software during maintenance requires that software engineers understand how various components of a design interact. People usually think of software maintenance as beginning when the product is delivered to the client. While this is formally true, in fact decisions that affect the maintenance of the product are made from the earliest stage of design.

Software maintenance is classified into four types: corrective, adaptive, perfective and preventive. Corrective maintenance refers to fixing a program. Adaptive maintenance refers to modifications that adapt to changes in the data environment, such as new product codes or new file organization or changes in the hardware of software environments. Perfective maintenance refers to enhancements: making the product better, faster, smaller, better documented, cleaner structured, with more functions or reports. The preventive maintenance is defined as the work that is done in order to try to prevent malfunctions or improve maintainability.

When a software system is not designed for maintenance, it exhibits a lack of stability under change. A modification in one part of the system has side effects that ripple throughout the system. Thus, the main challenges in software maintenance are to understand existing software and to make changes without introducing new bugs.

It is arguable that many object-oriented (OO) software systems are currently in use. It is also arguable that the growing popularity of OO programming languages, such as Java, as well as the increasing number of software development tools supporting the Unified Modelling Language (UML), encourages more OO systems to be developed at present and in the future. Hence it is important that those systems are maintained effectively and efficiently. A software maintainability prediction model enables organizations to predict maintainability of a software system and assists them with managing maintenance resources.

In addition, if an accurate maintainability prediction model is available for a software system, a defensive design can be adopted. This would minimize, or at least reduce future maintenance effort of the system. Maintainability of a software system can be measured in different ways. Maintainability could be measured as the number of changes made to the code during a maintenance period or be measured as effort to make those changes. The predictive model is called a maintenance effort prediction model if maintainability is measured as effort. Unfortunately, the number of software maintainability prediction models including maintenance effort prediction models, is currently very small in the literature.

In this research work, we developed a new maintainability prediction model for an object-oriented software system based on the recently introduced learning algorithm called Sensitivity Based Linear Learning Method (SBLLM). It is a learning technique for two-layer feed forward neural networks based on sensitivity analysis, which uses a linear training algorithm for each of the two layers. In theory, this algorithm tends to provide good generalization performance at extremely fast learning speed. The experimental results, found in literatures, based on a few artificial and real benchmark function approximation and classification problems including very large complex applications, and particularly the empirical results from this study, demonstrated that the SBLLM can produce good generalization performance in most cases and can learn thousands of times faster than conventional popular learning algorithms for feed-forward neural networks.

Despite the importance of software maintenance, little work has been done as regards developing predictive models for software maintainability, particularly object-oriented software system ,which is evident in the fewer number of software maintainability prediction models, that are currently found in the literature.

In view of this, we have developed a new maintainability prediction model for an object-oriented software system based on the recently introduced learning algorithm called Sensitivity Based Linear Learning Method. Implementation was carried out on representative datasets related to the target systems. Furthermore, we performed comparative analysis between our model and the models presented in, Koten (2006), which include Regression-Based and Bayesian Network Based models, in terms of their performance measures values, as recommended in the literatures.

Furthermore, the usefulness of the SBLLMs in the area of software engineering and, in particular, maintainability prediction for an object-oriented software system, has been made clearer by describing both the steps and the use of SBLLM as an artificial intelligence modeling

approach for predicting the maintainability of object-oriented software system.

The rest of this paper is organized as follows. Section 2 contains review of related earlier works. Section 3 discusses Sample predictive modelling techniques, and also describes the main modelling technique used: SBLLM. Section 4 presents the OO software data sets and the metrics used in our study and their descriptions. Section 5 model evaluation that include model validation approach and prediction accuracy measures used. Section 6 contains empirical results, comparison with other models and discussions. Section 7 concludes the paper.

2. Related Work

Several object oriented software maintainability prediction models were developed of recent; and they are mostly characterized by low prediction accuracies Lucia et al. (2005). Regression techniques have been thoroughly utilized by Li and Henry (1993), Fioravanti and Nesi (2001), and Misra(2005) to predict maintainability of object oriented software systems. Some recent work have been done using artificial neural networks and some other artificial intelligence techniques such as Bayesian Belief Networks (BBN), van and Gray (2006), and Multivariate adaptive regression splines (MARS), Zhou and Leung (2007).

Variants of artificial neural networks were also employed in predicting the maintainability effort of object oriented software systems. Feed forward neural network and General Regression neural network (GRNN) were used by Quah et al. (2003) to predict the maintainability effort for object oriented software systems using object oriented metrics. On the other hand, back-propagation multilayer perceptron (BP-MLP) have been used by Mahaweerawat, Sophatsathit, Lursinsap Musilek (2003) to predict faulty classes in object oriented software. In the same research work, they used radial basis function networks (RBF) to predict the type of fault a faulty class has.

Bayesian Belief Networks (BBN) was first suggested as a novel approach for software quality prediction by Fenton et al. (2002) and Fenton and Neil (1999, 2000). They build their conjecture based on Bayesian Belief Networks' ability in handling uncertainties, incorporating expert knowledge, and modeling the complex relationships among variables. However, a number of researchers, have pointed out several limitations of Bayesian Belief Networks when they are applied as a model for object oriented software quality and maintainability perdition Ma et al. (2006), Weaver (2003), and Yu et al. (2002). Recently, a special type of Bayesian Belief Networks called Naïve-Bayes classifier was used by van and Gray (2006) to implement a Bayesian-Belief-Networks-based software maintainability prediction model. Although their results showed that their model give better results than regression-based techniques for some datasets, the model is still inferior to regression-based techniques for some other datasets.

3. Sample Modeling Techniques

3.1 Regression Based Models

Regression models are used to predict one variable from one or more other variables. Regression models provide the scientist with a powerful tool, allowing predictions about past, present, or future events to be made with information about past or present events.

3.1.1 Multiple Linear Regression Model

Multiple linear regression attempts to model the relationship between two or more explanatory variables and a response variable by fitting a linear equation to observed data. Every value of the independent variable x is associated with a value of the dependent variable y.

The regression line for p explanatory variables $x_1, x_2, \ldots\ldots, x_p$ is defined to be $\mu_y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots. + \beta_p x_p$. This line describes how the mean response μ_y changes with the explanatory variables. The observed values for y vary about their means μ_y and are assumed to have the same standard deviation σ. Formally, the model for multiple linear regression given n observations is

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots. + \beta_p x_{ip} + \varepsilon_i$$

for i = 1,2,....n

Where ε_i is notation for model deviation.

One approach to simplifying multiple regression equations is the stepwise procedures. These include forward selection, backwards elimination, and stepwise regression. They add or remove variables one-at-a-time until some stopping rule is satisfied.

Forward Selection: Forward selection starts with an empty model. The variable that has the smallest P value when it is the only predictor in the regression equation is placed in the model. Each subsequent step adds the variable that has the smallest P value in the presence of the predictors already in the equation. Variables are added one-at-a-time as long as their P values are small enough, typically less than 0.05 or 0.10.

Backward Elimination: It starts with all of the predictors in the model. The variable that is least significant that is, the one with the largest P value is removed and the model is refitted. Each subsequent step removes the least significant variable in the model until all remaining variables have individual P values smaller than some value, such as 0.05 or 0.10.

Stepwise regression This approach is similar to forward selection except that variables are removed from the model if they become non significant as other predictors are added.

Backwards Elimination: has an advantage over forward selection and stepwise regression because it is possible for a set of variables to have considerable predictive capability rather than any individual subset. Forward selection and stepwise regression will fail to identify them because sometimes variables don't predict well individually and Backward elimination starts with everything in the model, so their joint predictive capability will be seen.

3.2 Bayesian Networks

A Bayesian network consists of nodes interconnected by the directed links forming directed acyclic graph. In this graph, nodes represent random variables (RVs) and links correspond to direct probabilistic influences. The RVs correspond to important attributes of the modeled system which exemplifying the system's behavior. Directed connection between the two nodes indicates a casual effect between RVs which associated with these nodes.

The structure of directed acyclic graph states that each node is independent of all its non descendants conditioned on its parent nodes. In other words, the Bayesian Network represents the conditional probability distribution P(Y/ X1,...,Xn) which is used to quantify the strength of variables Xi on the variable Y, Nodes Xi are called the parents of Y and Y is called a child of each Xi. This should be noted that outcomes of the events for the variables Xi have an influence on the outcome of the event Y.

3.3 Sensitivity Based Linear Learning Method (SBLLM)

Castillo et al (2006), proposed a new learning scheme in order to both speed up and avoid local minima convergence of the existing backpropagation learning

technique. This new learning strategy is called the Sensitivity Based Linear Learning (SBLLM) scheme. It is a learning technique for two-layer feedforward neural networks based on sensitivity analysis, which uses a linear training algorithm for each of the two layers. First, random values are assigned to the outputs of the first layer; later, these initial values are updated based on sensitivity formulas, which use the weights in each of the layers; the process is repeated until convergence. Since these weights are learnt solving a linear system of equations, there is an important saving in computational time. The method also gives the local sensitivities of the least square errors with respect to input and output data, with no extra computational cost, because the necessary information becomes available without extra calculations. This new scheme can also be used to provide an initial set of weights, which significantly improves the behavior of other learning algorithms. The full theoretical basis for SBLLM and its performance has been demonstrated in Castillo et al (2006), which contained its application to several learning problems examples in which it is compared with several learning algorithms and well known data sets. The results have shown a learning speed generally faster than other existing methods. In addition, it can be used as an initialization tool for other well known methods with significant improvements.

Sensitivity analysis is a very useful technique for deriving how and how much the solution to a given problem depends on data, see Castillo et al., 1997, 1999, 2000 and the references therein for more details. However, in Castillo et al (2006) it was shown that sensitivity formulas can also be used as a novel supervised learning algorithm for two-layer feedforward neural networks that presents a high convergence speed. Generally, SBLLM process is based on the use of the sensitivities of each layer's parameters with respect to its inputs and outputs and also on the use of independent systems of linear equations for each layer to obtain the optimal values of its parameters. In addition, it gives the sensitivities of the sum of squared errors with respect to the input and output data.

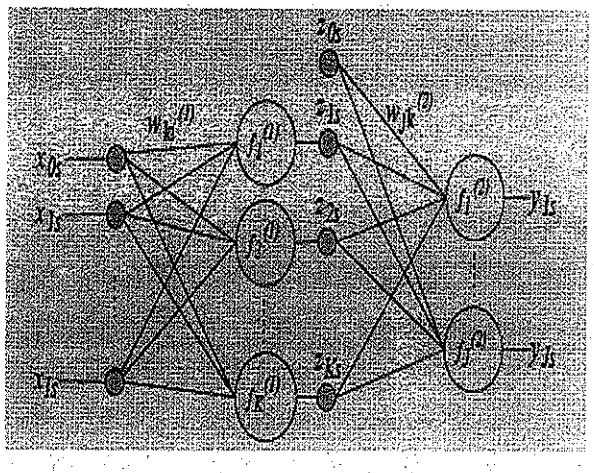


Figure 1: Two-layer feedforward neural network.

3.3.1 How Sensitivity Based Linear Learning Method (SBLLM) Works

Consider the two-layer feedforward neural network in Figure 1, where I is the number of inputs, J the number of outputs, K the number of hidden units, x0s =1, z0s =1, S the number of data samples and the superscripts (1) and (2) are used to refer to the first and second layer, respectively. This network can be considered to be composed of two one-layer neural networks as it is shown in Figure 2. For this one layer neural network, to learn the weights w_{ji}, one can minimize the sum of squared errors before the nonlinear activation functions, Castillo et al. (2002), that is,

$$Q=\sum_{s=1}^{S}\sum_{j=1}^{J}\varepsilon_{js}^{2}=\sum_{s=1}^{S}\sum_{j=1}^{J}\Big(\sum_{i=0}^{I}w_{ji}x_{is}-f_j^{-1}(y_{js})\Big)^2 \qquad (1)$$

this leads to the system of equations :

$$\sum_{i=0}^{I}A_{pi}w_{ji}=b_{pj};\ p=0,1,\ldots,I;\ \forall j, \qquad (2)$$

where

$$A_{pi} = \sum_{s=1}^{S} x_{is} x_{ps}; p = 0,1,\ldots,I; \forall i,$$

$$b_{pj} = \sum_{s=1}^{S} f_j^{-1}(y_{js}) x_{ps}; p = 0,1,\ldots,I; \forall j,$$

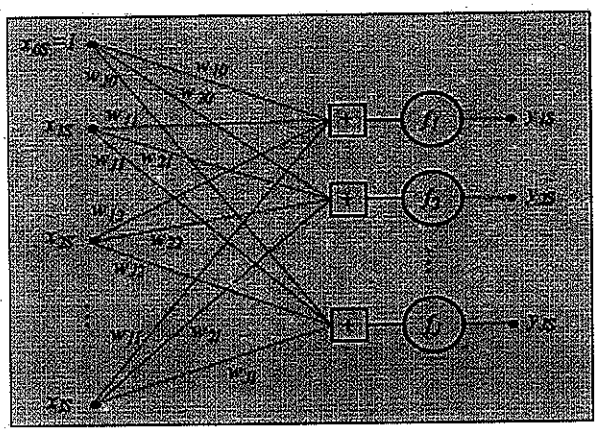


Figure 2 : One-layer feedforward neural network.

Therefore, assuming that the intermediate layer outputs z are known, using equation (1), a new cost function for the two-layer feedforward neural network in Figure 1 is defined as:

$$Q(z) = Q^{(1)}(z) + Q^{(2)}(z) =$$
$$= \sum_{s=1}^{S} \left[\sum_{k=1}^{K} \left(\sum_{i=0}^{I} w_{ki}^{(1)} x_{is} - f_k^{(1)^{-1}}(z_{ks}) \right) \right]$$

Thus, using the outputs z_k's we can learn, for each layer independently, the weights $w^{(1)}_{ki}$ and $w^{(2)}_{jk}$ by solving the corresponding linear system of equations (2). For the neural network shown in Figure 1, according to Castillo et al.(2001,2004, and 2006), , the sensitivities of the new cost function, Q, with respect to the output and input data can be obtained as:

$$\frac{\partial Q}{\partial y_{pq}} = \frac{2\left(\sum_{i=0}^{I} w_{pi} x_{iq} - f_p^{-1}(y_{pq}) \right)}{f_p'(y_{pq})}; \forall p,q$$

$$\frac{\partial Q}{\partial x_{pq}} = 2\sum_{j=1}^{J} \left(\sum_{i=0}^{I} w_{ji} x_{iq} - f_j^{-1}(y_{jq}) \right) w_{jp}; \forall p,q$$

Thus, the sensitivities with respect to z_{ks} for the two - layer feedforward neural network in Figure-1 are calculated as:

$$\frac{\partial Q}{\partial z_{ks}} = \frac{\partial Q^{(1)}}{\partial z_{ks}} + \frac{\partial Q^{(2)}}{\partial z_{ks}} =$$

$$= -\frac{2\left(\sum_{i=0}^{I} w_{ki}^{(1)} x_{is} - f_k^{(1)^{-1}}(z_{ks}) \right)}{f_k'^{(1)}(z_{ks})}$$
$$+2\sum_{j=1}^{J} \left(\sum_{r=0}^{K} w_{jr}^{(2)} z_{rs} - f_j^{(2)^{-1}}(y_{js}) \right) w_{jk}^{(2)}$$

with k = 1, . . . ,K, as z_{0s} = 1,"s. After this, the values of the intermediate outputs z are modified using the Taylor series approximation:

$$Q(z + \nabla z) = Q(z) + \sum_{k=1}^{K} \sum_{s=1}^{S} \frac{\partial Q(z)}{\partial z_{ks}} \Delta z_{ks} \approx 0, \quad (3)$$

this leads to the following increments

$$\Delta z = -\rho \frac{Q(z)}{\|\nabla Q\|^2} \nabla Q,$$

Where ρ is a relaxation factor or step size. The Sensitivity-Based Linear Learning scheme is summarized in the following algorithmic steps.

3.3.2 SBLLM Learning Process

The training algorithm of the SBLLM technique can be summarized in the following algorithmic steps:

Input- The inputs to the system, which is the available or simulated data (training) set (input, xis, and desired data, yjs), two threshold errors (ε and ε') to control both convergence and a step size ρ.

Output- The output results of the SBLLM system are the weights of the two layers and the sensitivities of the sum of squared errors with respect to input and output data.

Step 0: Initialization. Assign to the outputs of the intermediate layer the output associated with some random weights w(1)(0) plus a small random error, that is,

$$z_{ks} = f_k^{(1)}\Big(\sum_{i=0}^{I} w_{ki}^{(1)}(0)x_{is}\Big) + \varepsilon_{ks}; \varepsilon_{ks} \to U(-\eta,\eta);$$
$$k = 1,...K,$$

where η is a small number, and initialize the sum of squared errors $(Q)_{previous}$ and mean-squared errors $(MSE)_{previous}$ to some large number, where MSE measures the error between the obtained and the desired output.

Step 1: Sub-problem solution. Learn the weights of layers 1 and 2 and the associated sensitivities solving the corresponding systems of equations, that is,

$$\sum_{i=0}^{I} A_{pi}^{(1)} w_{ki}^{(1)} = b_{pk}^{(1)}$$

$$\sum_{k=0}^{K} A_{qk}^{(2)} w_{jk}^{(2)} = b_{qj}^{(2)}$$

Where

$$A_{pi}^{(1)} = \sum_{s=1}^{S} x_{is}x_{ps}; b_{pk}^{(1)} = \sum_{s=1}^{S} f_k^{(1)^{-1}}(z_{ps})x_{ps};$$
$$p = 0,1,.....,I; k = 1,2,.....,K,$$

and

$$A_{qk}^{(2)} = \sum_{s=1}^{S} z_{ks}z_{qs}; b_{qj}^{(2)} =$$
$$\sum_{s=1}^{S} f_k^{(2)^{-1}}(y_{js})z_{qs}; q = 0,1,.....,K; \forall j.$$

Step 2: Evaluate the sum of squared errors. Evaluate Q using

$$Q(z) = Q^{(1)}(z) + Q^{(2)}(z) =$$
$$= \sum_{s=1}^{S}\Big[\sum_{k=1}^{K}\Big(\sum_{i=0}^{I} w_{ki}^{(1)}x_{is} - f_k^{(1)^{-1}}(z_{ks})\Big)^2$$
$$+\sum_{j=1}^{J}\Big(\sum_{k=0}^{K} w_{jk}^{(2)}z_{ks} - f_k^{(2)^{-1}}(y_{js})\Big)^2\Big],$$

and evaluate also the MSE.

Step 3: Convergence checking. If $|Q - Q_{previous}| < \varepsilon$ or $|MSE_{previous} - MSE| <$ stop and return the weights and the sensitivities. Otherwise, continue with Step 4.

Step 4: Check improvement of Q. If $Q > Q_{previous}$ reduce the value of ρ, that is, $\rho = \rho / 2$ and return to the previous position, that is, restore the weights, z = zprevious, Q = Qprevious and go to Step 5. Otherwise, store the values of Q and z, that is, Qprevious = Q, MSEprevious =MSE and zprevious = z and obtain the sensitivities using:

$$\frac{\partial Q}{\partial z_{ks}} = -\frac{2\Big(\sum_{i=0}^{I} w_{ki}^{(1)}x_{is} - f_k^{(1)^{-1}}(z_{ks})\Big)}{f_k^{(1)}(z_{ks})} +$$
$$2\sum_{j=1}^{J}\Big(\sum_{r=0}^{K} w_{jr}^{(2)}z_{rs} - f_j^{(2)^{-1}}(y_{js})\Big)w_{jk}^{(2)}; k = 1,....,K.$$

Step 5: Update intermediate outputs. Using the Taylor series approximation in equation (Eq.s3), update the intermediate outputs as

$$z = z - \rho\frac{Q(z)}{\|\nabla Q\|^2}\nabla Q,$$

and go to Step 1.

3.4.3 Advantages of SBLLM

SBLLM offers an interesting combination of speed, reliability and simplicity. In addition, based on the results obtained from the real-world experiments using the SBLLM learning algorithm, there are four main advantages of the SBLLM that can be summarized as follows, Castillo et al. (2006):

1. High speed in reaching the minimum error: It was demonstrated in Castillo et al 2006, that in all cases the SBLLM obtains its minimum MSE (MSE_{min}) just before the first four iterations and also sooner than the rest of the algorithms examined together. SBLLM gets its minimum error in an epoch for which the other algorithms are far from similar MSE values.

2. Good performance: It can be deduced that not only that SBLLM stabilizes soon, but also the minimum MSE that it reaches is quite good and comparable to that obtained by the second order methods. Other methods compared to it never succeeded in attaining this minimum MSE (MSE_{min}) before the maximum number of epochs.

3. Homogeneous behavior: The SBLLM learning curve stabilizes soon as it is demonstrated in Castillo et al. (2006). The SBLLM behaves homogeneously not only if we consider just the end of the learning process, but also during the whole process, in such a way that very similar learning curves where obtained for all iterations of different experiments.

4. SBLLM as Initialization Method: The other good aspect of SBLLM is that it has been used as initialization method in conjunction with other learning algorithm often with better and accurate results. In this case it always achieves a faster convergence speed, obtains a very good initial point, and thus a very low MSE in a few epochs of training. In addition, the SBLLM helps the learning algorithms to obtain a more homogeneous Mean square error (MSE) at the end of the training process. Therefore, when SBLLM is used as an initialization method, it significantly improves the performance of a learning algorithm.

4. Data Sets

In this work, we made use of OO software datasets published by Li and Henry (1993). In this section, we describe the data set used for this study. We first introduce the metrics under study and then give some statistical analysis of the metrics that were investigated.

4.1. Studied Metrics

This study makes use of two OO software data sets published by Li and Henry (1993). These metric data were collected from a total of 110 classes in two OO software systems The first data set, UIMS, contains the metric data of 39 classes collected from a user interface management system (UIMS). The second data set, QUES, contains the metric data of 71 classes collected from a quality evaluation system (QUES). Both systems were implemented in Ada.

The datasets consist of five C&K metrics: DIT, NOC, RFC, LCOM and WMC, and four L&H metrics: MPC, DAC, NOM and SIZE2, as well as SIZE1, which is a traditional lines of code size metric. Maintainability was measured in CHANGE metric by counting the number of lines in the code, which were changed during a three-year maintenance period. Neither UIMS nor QUES datasets contain actual maintenance effort data. The description of each metric is given in the table below:

Table 1: Description Of Metrics

Metric	Description
WMC (Weighted methods per class)	The sum of McCabe's cyclomatic complexity of all local methods in a given class
DIT (Depth of inheritance tree)	The length of the longest path from a given class to the root in the inheritance hierarchy
RFC (Response for a class)	The number of methods that can potentially be executed in response to a message being received by an object of a given class
NOC (Number of children)	The number of classes that directly inherit from a given class . i.e. number of direct sub-classes that the class has
LCOM (Lack of cohesion in methods)	The number of pairs of local methods in a given class using no attribute in common . number of disjoint sets of local methods, i.e. number of sets of local methods that do not interact with each other, in the class
MPC (Message -passing coupling)	The number of send statements defined in a given class
DAC (Data abstraction coupling)	The number of abstract data types defined in a given class
NOM (Number of methods)	The number of methods implemented within a given class
SIZE1 (Lines of code)	The number of semicolons in a given class
SIZE2 (Number of properties)	The total number of attributes and the number of local methods in a given class
CHANGE (Number of lines changed in the class)	Insertion and deletion are independently counted as 1, change of the contents is counted as 2

DIT, NOC, RFC, LCOM, WMC, MPC, DAC, NOM, SIZE2, and SIZE1 are the features that are combined and made used of to predict the attribute change. QUES data set has 71 sample cases, whereas UIMS has 39 sample cases

4.2 Characteristics of the Datasets

Table 2 : Descriptive Statistics Of The UIMS Data Set

Metric	Maximum	75%	Median	25%	Minimum	Mean	Standard deviation	Skewness
WMC	69	12	5	1	0	11.38	15.90	2.03
DIT	4	3	2	2	0	2.15	0.90	- 0.54
RFC	101	30	17	11	2	23.21	20.19	2.00
NOC	8	1	0	0	0	0.95	2.01	2.24
LCOM	31	8	6	4	1	7.49	6.11	2.49
MPC	12	6	3	1	1	4.33	3.41	0.731
DAC	21	3	1	0	0	2.41	4.00	3.33
NOM	40	13	7	6	1	11.38	10.21	1.67
SIZE1	439	131	74	27	4	106.44	114.65	1.71
SIZE2	61	16	9	6	1	13.97	13.47	1.89
CHANGE	289	39	18	10	2	46.82	71.89	2.29

Table 3: Descriptive Statistics of The QUES Data Set

Metric	Maximum	75%	Median	25%	Minimum	Mean	Standard deviation	Skewness
WMC	83	22	9	2	1	14.96	17.06	1.77
DIT	4	2	2	2	0	1.92	0.53	- 0.10
RFC	156	62	40	34	17	54.44	32.62	1.62
NOC	0	0	NA	0	0	0	0.00	NA
LCOM	33	14	5	4	3	9.18	7.34	1.35
MPC	42	21	17	12	2	17.75	8.33	0.88
DAC	25	4	2	1	0	3.44	3.91	2.99
NOM	57	21	6	5	4	13.41	12.00	1.39
SIZE1	1009	333	211	172	115	275.58	171.60	2.11
SIZE2	82	25	10	7	4	18.03	15.21	1.71
CHANGE	217	85	52	35	6	64.23	43.13	1.36

5. Model Evaluation

5.1 Model Validation Approach

The available data set, for each data set, were divided into two parts. One part was used as a training set, for constructing a maintainability prediction model. The other part was used for testing to determine the prediction ability of the developed model. Although there are many different ways to split a given dataset, we have chosen to use the stratify sampling approach in breaking the datasets due to its ability to break data randomly with a resultant balanced division based on the supplied percentage. The division, for instance could be 70% for training set and 30% for testing set. In this work, we selected 70% of the data for building the model (internal validation) and 30% of the data for testing/ validation (external validation or cross-validation criterion). We repeat both internal and external validation processes for 1000 times to have a fair partition through the entire process operations.

We also evaluate and compare our developed model with other OO software maintainability prediction models,

sited earlier, quantitatively, using the following prediction accuracy measures recommended in the literatures: absolute residual (Ab.Res.), the magnitude of relative error (MRE) and the proportion of the predicted values that have MRE less than or equal to a specified value suggested in the literatures (pred measures). Details of all these measures of performance will be provided shortly.

5.2 Prediction Accuracy Measures

In this paper, we compared the software maintainability prediction models using the following prediction accuracy measures: absolute residual (Abs Res), the magnitude of relative error (MRE) and Pred measures.

The Ab.Res. is the absolute value of residual evaluated by:

Ab.Res. = abs (actual value - predicted value)

In this paper, the sum of the absolute residuals (Sum Ab.Res.), the median of the absolute residuals (Med.Ab.Res.) and the standard deviation of the absolute residuals (SD Ab.Res.) are used. The Sum Ab.Res. measures the total residuals over the dataset. The Med.Ab.Res. measures the central tendency of the residual distribution. The Med.Ab.Res. is chosen to be a measure of the central tendency because the residual distribution is usually skewed in software datasets. The SD Ab.Res. measures the dispersion of the residual distribution.

MRE is a normalized measure of the discrepancy between actual values and predicted values given by

MRE = abs (actual value - predicted value) / actual value

The Max.MRE measures the maximum relative discrepancy, which is equivalent to the maximum error relative to the actual effort in the prediction. The mean of MRE, the mean magnitude of relative error (MMRE):

$$MMRE = \frac{1}{n}\sum_{i=1}^{n} MRE_i$$

According to Fenton and Pfleeger (1997), Pred is a measure of what proportion of the predicted values have MRE less than or equal to a specified value, given by:

Pred (q) = k / n

where q is the specified value, k is the number of cases whose MRE is less than or equal to q and n is the total number of cases in the dataset.

According to Conte and Dunsmore (1986), and MacDonell(1997), in order for an effort prediction model to be considered accurate, MMRE $\leq$ 0.25 and/or either pred(0.25) $\geq$ 0.75 or pred(0.30) $\geq$ 0.70 . These are the suggested criteria in literature as far as effort prediction is concerned.

6. Empirical Results And Comparison

Below are tables and figures showing the results of our newly developed model in comparison to the other earlier models used on the same datasets.

6.1 Results from QUES dataset

Table 3 shows the values of the prediction accuracy measures achieved by each of the maintainability prediction models for the QUES dataset. In order for an effort prediction model to be considered accurate, either MMRE $\leq$ 0.25 and/or either pred(0.25) $\geq$ 0.75 or pred(0.30) $\geq$ 0.70, needed to be achieved, Conte and Dunsmore (1986), and MacDonell(1997) .Hence the closer a model's prediction accuracy measure value is to these baseline values, the better. Since Table 3 shows that the SBLLM model has achieved MMRE value of 0.348, the pred(0.25) value of 0.5 and the pred(0.30) value of

0.56. It is clear from these, that the SBLLM model is the only one that is very close to the required values for all the three essential prediction measures, hence it is the best among all the presented models. It outperform all the other model in terms of all the predictive measures used.

In comparison with the UIMS dataset, the performance on QUES dataset is far better than that on UIMS. This indicates that the performance of the SBLLM models may vary depending on the characteristics of the dataset and/ or depending on what prediction accuracy measure is used.

Table 4 : Prediction Accuracy For The QUES Dataset

Model	Max. MRE	MMRE	Pred (0.25)	Pred (0.30)	Sum Ab.Res.	Med. Ab.Res.	SD Ab.Res.
Bayesian network	1.592	0.452	0.391	0.430	686.610	17.560	31.506
Regression Tree	2.104	0.493	0.352	0.383	615.543	19.809	25.400
Backward Elimination	1.418	0.403	0.396	0.461	507.984	17.396	19.696
Stepwise Selection	1.471	0.392	0.422	0.500	498.675	16.726	20.267
SBLLM	1.713	0.348	0.5	0.56	778.437	11.274	16.258

6.2 Results From UIMS Dataset

Table 3 shows the values of the prediction accuracy measures achieved by each of the maintainability prediction models for the UIMS dataset. From the results presented, the SBLLM model has achieved the MMRE value of 1.966, the pred (0.25) value of 0.179 and the pred (0.30) value of 0.25. These values compete favorably among all the five models presented. Specifically in term of Sum.Abs. value, it is the best among all the models and we can see that there is strong evidence that the SBLLM model's value is significantly lower and thus, better than those of the other models. In term of pred(0.30), it is the second best model after Bayesian network. In addition, it is also the best in term of Med. Ab.Res, and SD. Ab.Res.

Though the performance of SBLLM on UIMS dataset is low compared to its performance on QUES dataset, yet its performance compared to other models on same dataset is competitive and encouraging.

Table 5 : Prediction Accuracy for the UIMS Dataset

Model	Max. MRE	MMRE	Pred (0.25)	Pred (0.30)	Sum Ab.Res.	Med. Ab.Res	SD Ab.Res.
Bayesian network	7.039	0.972	0.446	0.469	362.300	10.550	46.652
Regression Tree	9.056	1.538	0.200	0.208	532.191	10.988	63.472
Backward Elimination	11.890	2.586	0.215	0.223	538.702	20.867	53.298
Stepwise Selection	12.631	2.473	0.177	0.215	500.762	15.749	54.114
SBLLM	13.053	1.966	0.179	0.25	240.583	7.7452	18.095

6.3 Discussion

With the exception of SBLLM that has values closer to satisfying the stated criteria, particularly on QUES dataset, non of the other prediction models presented get closer to satistisfying any of the criteria of an accurate prediction model cited earlier. However, it is reported that prediction accuracy of software maintenance effort prediction models are often low and thus, it is very difficult to satisfy the criteria, Lucia et al. (2005).

Thus, we are concluding that SBLLM model presented in this paper can predict maintainability of the OO software systems reasonably well to an acceptable degree. This submission of ours is as a result of the fact that only SBLLM model has been able to consistently perform better by having values closer to satisfying the criteria laid down in literature particularly on the QUES dataset.. For UIMS datasets, whenever the SBLLM model's prediction accuracy has not been as good as the other models, it has been reasonably close. In terms of absolute residuals, SBLLM is better than other models for both datasets.

For better visualization of results, the prediction accuracy measures for both UIMS and QUES datasets are depicted in figures 5 and 6 respectively.

7. Conclusion

An SBLLM OO software maintainability prediction model has been constructed using the OO software metric data in Li and Henry datasets, Li and Henry (1993). The prediction accuracy of the model is evaluated and compared with the bayelsian network model, regression tree model and the multiple linear regression models using the prediction accuracy measures: the absolute residuals, MRE and pred measures. The results indicate that SBLLM model can reliably predict maintainability of the OO software systems. The SBLLM model has achieved significantly better prediction accuracy, than the other models, particularly on QUES dataset. Also, for UIMS datasets, whenever the SBLLM model's prediction accuracy measure is not better than the best among the other models, it has been reasonably competitive against the best models.

Therefore, we are concluding that the prediction accuracy of the SBLLM model is better than, or at least, is competitive against the Bayesian network model and the regression based models. These outcomes have confirmed that SBLLM is indeed a useful modeling technique for software maintainability prediction, although further studies are required to realize its full potentials.

The results in this paper also suggest that the prediction accuracy of the SBLLM model may vary depending on the characteristics of dataset and/or the prediction accuracy measure used. This provides an interesting direction for future studies.

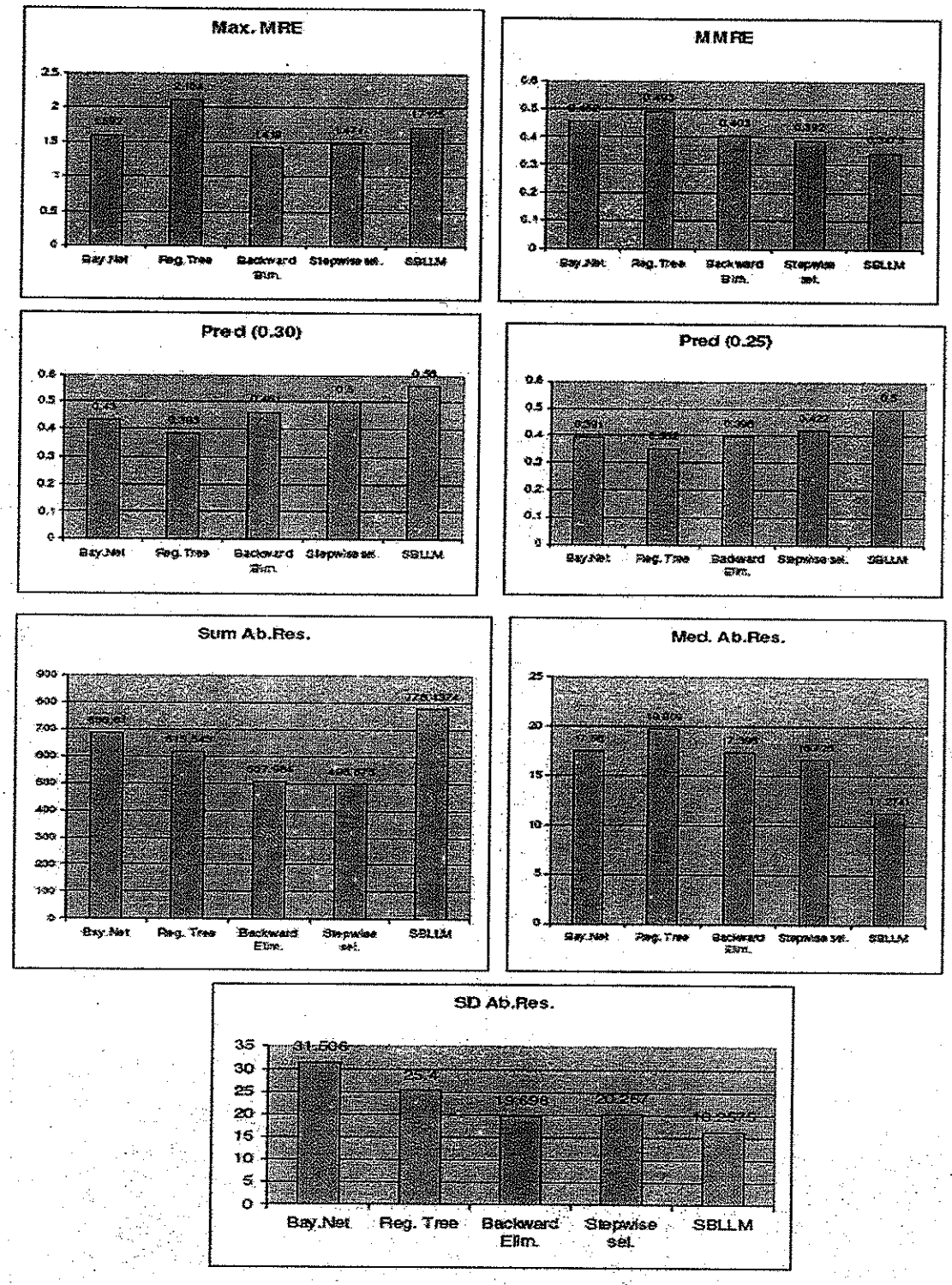


Figure 5 : Charts Depicting The Prediction Accuracy For The QUES Dataset

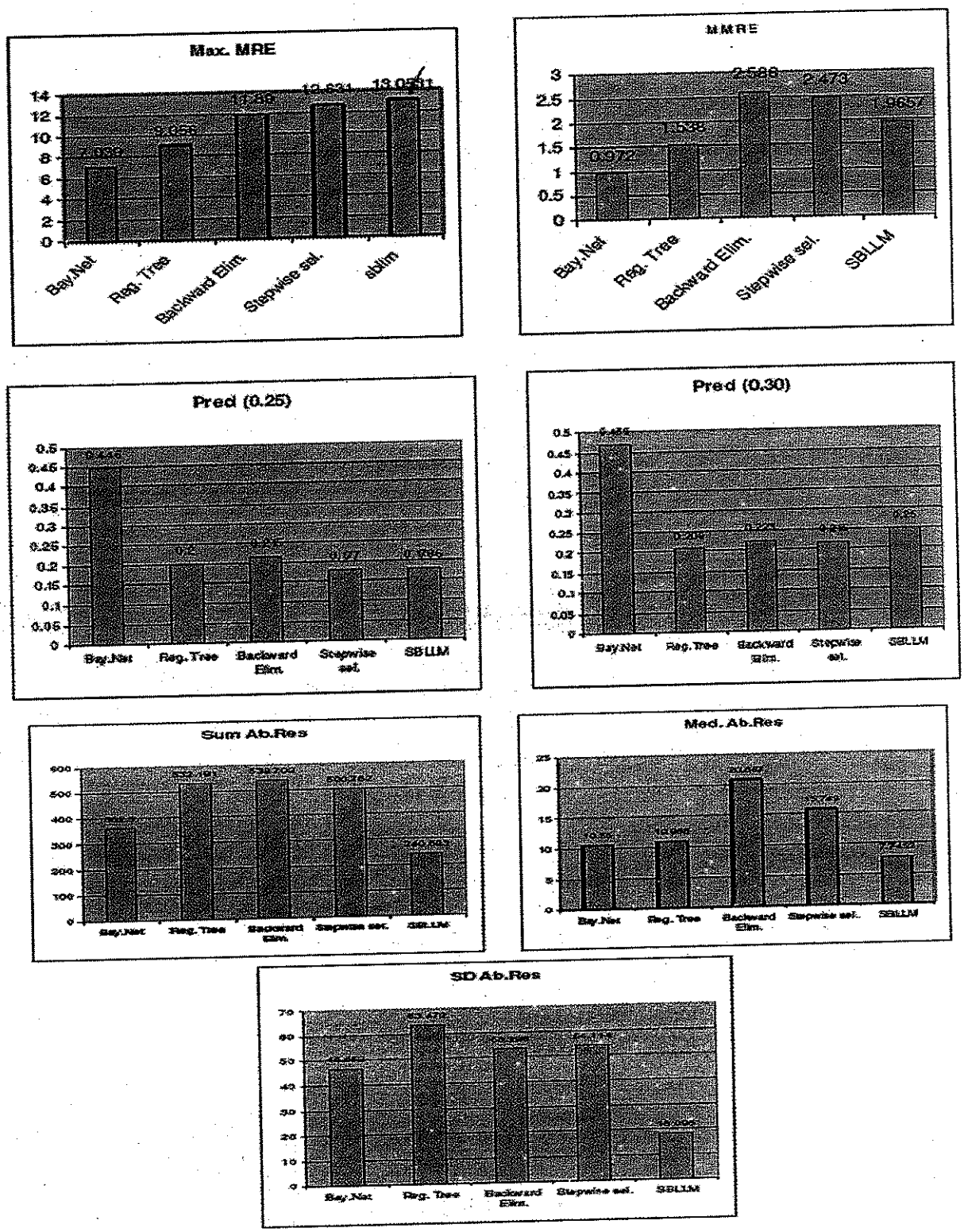


Figure 6 : Charts Depicting The Prediction Accuracy For The UIMS Dataset

References

[1] E. Castillo, C. Castillo A. Conejo and R. M´ýnguez and D. Ortigosa, *"A perturbation approach to sensitivity analysis in nonlinear programming"*, Journal of Optimization Theory and Applications, 128(1):49–74, 2006.

[2] Castillo. A. S, Hadi. A, Conejo and A. Fern´andez-Canteli, *"A general method for local sensitivity analysis with application to regression models and other optimization problems"*, Technometrics, 46(4):430–445, 2004.

[3] Castillo. C, Castillo. A, Conejo and R. M´ýnguez and D. Ortigosa, *"A perturbation approach to sensitivity analysis in nonlinear programming"*, Journal of Optimization Theory and Applications, 128(1):49–74, 2006.

[4] Castillo. A, Conejo. P, Pedregal. R, Garc´ýa and N. Alguacil, *"Building and Solving Mathematical Programming Models in Engineering and Science"*, John Wiley & Sons Inc., New York., 2001.

[5] Castillo, A. Cobo, J. M. Guti´errez, and R. E. Pruneda, *"Working with differential, functional and difference equations using functional networks"*, Applied Mathematical Modelling, 23(2):89–107, 1999.

[6] Enrique Castillo, Bertha Guijarro-Berdi ¨nas, Oscar Fontenla-Romero, *"Amparo Alonso-Betanzos A Very Fast Learning Method for Neural Networks Based on Sensitivity Analysis"*, Journal of Machine Learning Research 7, 1159–1182, 2006.

[7] Castillo, J. M. Guti´errez, and A. Hadi, *"Sensitivity analysis in discrete bayesian networks"*, IEEETransactions on Systems, Man and Cybernetics, 26(7):412–423, 1997.

[8] Castillo, O. Fontenla-Romero, A. Alonso Betanzos and B. Guijarro-Berdi¨nas, *"A global optimum approach for one-layer neural networks"*, Neural Computation, 14(6):1429–1449, 2002.

[9] Chikako van Koten, Andrew Gray, *"An Application of Bayesian Network for Predicting Object-Oriented Software Maintainability"*, The Information Science, Discussion Paper Series, 2006.

[10] L.C. Briand, T. Langley and I. Wieczorek. A, *"replicated assessment and comparison of common software cost estimation modelling techniques"*, In Proceedings of the 22nd International Conference on Software Engineering (ICSE'00), PP 377–386, 2000.

[11] L.C. Briand and J. W¨ust, *"The impact of design properties on development cost in object-oriented systems"*, In Proceedings of the 7th International Software Metrics Symposium (METRICS'01), PP. 260–271, 2001.

[12] L.C. Briand and J. W¨ust, *"Modeling development effort in object-oriented systems using design properties"*, IEEE Transactions on Software Engineering, 27(11):963–986, 2001.

[13] N.E. Fenton and S.L. Pfleeger, *"Software Metrics:A Rigorous & Practical Approach"*, PWS Publishing Company, second edition, 1997.

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

[15] W. Li and S. Henry, *"Object-oriented metrics that predict maintainability"*, Journal of Systems and Software, 23:111–122, 1993.

[16] S.G. MacDonell, *"Establishing relationships between specification size and software process*

effort in case environment", Information and Software Technology, 39:35–45, 1997.

[17] L. Pickard, B. Kitchenham, and S. Linkman, *"An investigation of analysis techniques for software datasets"*, In Proceedings of the 6th International Software Metrics Symposium (METRICS'99), PP. 130–142, 1999.

[18] E. Stensrud, *"Alternative approaches to effort prediction of erp projects"*, Information and Software Technology, 43:413–423, 2001.

[19] F. Fioravanti and P. Nesi, *"Estimation and Prediction Metrics for Adaptive Maintenance Effort of Object-Oriented Systems"*, IEEE Transactions on Software Engineering, Vol. 27, No. 12, 2001, PP. 1062–1084.

[20] T.M. Khoshgoftaar, E.B. Allen, J.P. Hudepohl, and S.J. Aud, *"Applications of Neural Networks to Software Quality Modeling of a Very Large Telecommunications System"*, Trans. Neural Networks, Vol. 8, No. 4, PP. 902-909, 1997.

[21] N.E. Fenton and M. Neil, *"A Critique of Software Defect Prediction Research"*, IEEE Transactions on Software Engineering, Vol. 25, No. 5, PP. 675–689, 1999.

[22] Wei Li and S. Henry, *"Object-Oriented Metrics that Predict Maintainability"*, Journal of Systems and Software, PP.111-122, 1993.

[23] El Emam, W. Melo, C.M. Javam, *"The Prediction of Faulty Classes Using Object-Oriented Design Metrics"*, Journal of Systems and Software, Elsevier Science, PP. 63-75, 2001.

[24] Kamaldeep Kaur, Arvinder Kaur and Ruchika Malhotra, *"Alternative Methods to Rank the Impact of Object Oriented Metrics in Fault Prediction Modeling using Neural Networks"*, Proceedings Of World Academy Of Science, Engineering And Technology, Vol. 13, PP. 207-212, 2006.

[25] S.Kanmani, V. Rhymend Uthariaraj, V. Sankaranarayanan and P. Thambidurai, *"Object Oriented Software Quality Prediction Using General Regression Neural Networks"*, ACM SIGSOFT Software Engineering Notes, Vol. 29, 2004, PP. 1-5, 2004.

[26] N.E. Fenton, P. Krause and M. Neil, *"Software Measurement: Uncertainty and Causal Modeling"*, IEEE Software, Vol. 10, No. 4, PP. 116-122, 2002.

[27] M. M. T. Thwin and T.-S. Quah, *"Application of Neural Networks for predicting Software Development faults using Object Oriented Design Metrics"*, Proceedings of the 9th International Conference on Neural Information Processing, PP. 2312 – 2316, November, 2002.

[28] van Koten. C, Gray. A.R, *"An Application of Bayesian Network for Predicting Object-Oriented Software Maintainability"*, Information and Software Technology , Vol. 48, No. 1, PP. 59–67, 2006.

[29] G G.-B. Huang and H. A. Babri, *"Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions"*, IEEE Transactions on Neural Networks, Vol. 9, No. 1, PP. 224—229, 1998.

[30] N.E. Fenton and S.L. Pfleeger, *"Software Metrics:A Rigorous & Practical Approach"*, PWS Publishing Company, second edition, 1997.

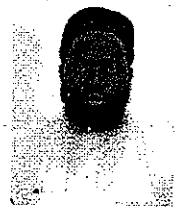
[31] S.D. Conte, H.E. Dunsmore and V.Y. Shen. *"Software Engineering Metrics and Models"*, Benjamin/Cummings Publishing Company, 1986.

[32] S.G. MacDonell, *"Establishing relationships between specification size and software process effort in case environment"*, Information and Software Technology, 39:35–45, 1997.

[33] A. De Lucia, E. Pompella and S. Stefanucci, *"Assessing effort estimation models for corrective maintenance through empirical studies"*, Information and Software Technology, 47:3–15, 2005.

[34] W. Li and S. Henry, *"Object-oriented metrics that predict maintainability"*, Journal of Systems and Software, 23:111–122, 1993.

Author's Biography



Sunday Olusanya Olatunji received the B.Sc. (Hons) Degree in Computer Science, Ondo State University (Now University of Ado Ekiti), Nigeria in 1999. He received M.Sc. Degree in Computer Science, University Of Ibadan, Nigeria in 2003. He received another M.Sc. Degree in Information and Computer Science, King Fahd University of Petroleum and Minerals (KFUPM), Saudi Arabia in 2008. He is currently pursuing his Phd in Computer Science. He has been a Lecturer in Computer Science Department, Ondo State University, Akungba Akoko, Nigeria, since 2001, where he is presently on study leave to obtain his Ph.D. He is a member of ACM and IEEE. He has participated in numerous research projects in KFUPM including those with ARAMCO oil and gas Company.