Chapter 13
A Metric Suite for Predicting Software Maintainability in Data Intensive Applications

Ruchika Malhotra and Anuradha Chug

Abstract Software maintainability is the vital aspect of software quality and defined as the ease with which modifications can be made once the software is delivered. Tracking the maintenance behaviour of a software product is very complex that is widely acknowledged by the researchers. Many research studies have empirically validated that the prediction of object oriented software maintainability can be achieved before actual operation of the software using design metrics proposed by Chidamber and Kemerer (C&K). However, the framework and reference architecture in which the software systems are being currently developed have changed dramatically in recent times due to the emergence of data warehouse and data mining field. In the prevailing scenario, certain deficiencies were discovered when C&K metric suite was evaluated for data intensive applications. In this study, we propose a new metric suite to overcome these deficiencies and redefine the relationship between design metrics with maintainability. The proposed metric suite is evaluated, analyzed and empirically validated using five proprietary software systems. The results show that the proposed metric suite is very effective for maintainability prediction of all software systems in general and for data intensive software systems in particular. The proposed metric suite may be significantly helpful to the developers in analyzing the maintainability of data intensive software systems before deploying them.

Keywords Data intensive applications · Empirical validation · Machine learning · Prediction models · Software design metric · Software maintainability

R. Malhotra
Department of Software Engineering, Delhi Technological University, Delhi, India
e-mail: ruchikamalhotra2004@yahoo.com

A. Chug (✉)
University School of Information and Communication Technology, Guru Gobind Singh Indraprastha University, Dwarka, New Delhi, India
e-mail: a_chug@yahoo.co.in

H. K. Kim et al. (eds.), *Transactions on Engineering Technologies*,
DOI: 10.1007/978-94-017-9115-1_13,

1 Introduction

Producing software which does not need change is not only impractical but also very uneconomical. The process of making changes in the software once it has been delivered to the customer is called software maintenance [1] and the ease with which it could be done is called as software maintainability [2]. The amount of resource, effort and time spent on it is much more than what is being spent on development. Thus, producing software that is easy to maintain may potentially save large costs [3]. Practitioners have suggested many ways to control the maintenance cost and one of them is to utilize the software design metrics and predict maintainability in the early phases of project development [3, 4]. Maintenance cost can be kept under control by accurate prediction of software maintainability due to many reasons such as:

(a) Productivity cost among projects can be compared.
(b) More effective planning of valuable resources can be done in advance.
(c) Major decision regarding staff allocation can be timely made.
(d) The threshold values of various metrics can be checked.
(e) Determinants of software quality can be enhanced.
(f) Practitioners are able to achieve optimized maintenance costs.

Various metrics have been proposed in the literatures which have significant impact on software maintainability. The main purpose of this study is 2-fold, firstly to review the role of C&K metrics suite for the prediction of software maintainability and secondly to propose a new suite of metrics with the induction of two new metrics which have larger impact on maintainability in highly data intensive applications. In order to achieve the goal, the data was collected from five proprietary software systems developed in Microsoft Visual Studio using C# language and based on object oriented (OO) methodologies with heavy use of databases for processing of each query. To measure the features of OO paradigm, C&K metric suite proposed by Chidamber and Kemerer [5] has been found to be a significant indicator of maintainability predictions in large number of studies [6–15]. We rely on the outcome of these studies and use C&K metric suite to capture the OO characteristics. There were two deficiencies found in this metric suite. First observation was the same as noted by Li et al. [16] that it does not take into account the structural complexity of the software. To overcome this deficiency we added two metrics i.e. Maintainability Index (MI) proposed by Oman et al. [17, 18] and Cyclomatic Complexity (CC) proposed by McCabe [19]. The second and main deficiency found in the metric suite is on account of the amount of database handling. To overcome this deficiency, two new metrics were proposed and validated for the applications which heavily use databases. The proposed metrics are Number of Data Base Connections (NODBC) made each time for query processing and the Schema Complexity to Comment Ratio (SCCR) to measure the understandability of the databases. The metrics definition and collection is discussed in Sect. 3. Overall a set of ten metrics were considered as independent

variables in this study which included 06 from C&K metric suite and CC, MI, NODBC and SCCR while the dependent variable was the number of the changes made in the lines of source code. Two versions of each of the software systems were taken and analyzed to count the changes made in the new version with respect to the older version. Four different versions of Artificial Neural Network (ANN) i.e. Back Propagation Network (BPN), Kohonen Network (KN), Feed Forward Neural Network (FFNN) and General Regression Neural Networks (GRNN) are used for making the prediction model. Data analysis was performed using correlation coefficient to verify the findings. We found that the new proposed metrics suite is significantly related with dependent variable. It is also observed that maintainability predictions for the applications which heavily use databases were more precise and accurate using new metric suite. Univariate as well as multivariate analysis further confirmed the results and proved the significance of proposed metrics suite. By using the new proposed metrics suite software practitioners can considerably take decisions whether the developed application is maintainable or not, which would save the time and money for the organizations responsible for developing and deploying the customized software's for the customers to gain their better satisfaction in the industry. The rest of the chapter is organized as follows: Sect. 2 presents the related work and Sect. 3 introduces proposed metrics. Section 4 describes independent, dependent variables and data analysis. Section 5 describes the machine learning methods used in the predictions process. Section 6 presents results and analysis, Sect. 7 discusses threats to validity and finally Sect. 8 concludes the paper with future directions.

2 Related Work

The problem of predicting the maintainability of the software is widely acknowledged in the industry due to the subjectivity involved while trying to quantify it. Jorgensen [20] suggested that we can measure maintainability by measuring the change efforts during operations. Many empirical studies have been conducted to predict the software maintainability using various tools and processes at the time of designing an application [6–15]. Multiple Linear Regression (MLR) Model was used by Li and Henry [6] to predict maintenance effort and to ear marked those metrics which have strong impact on maintainability. Muthanna et al. [21] also used polynomial regression to establish the relationship between design level metrics using industrial software. The results using graph plots have shown that predicted values were quite close to the actual values. Dagpinar and Jahnke [9] also carried empirical study and recorded significant impact of direct coupling metrics and size on maintainability. Fioravanti and Nesi [22] presented a metric analysis to identify which metrics would be better ranked for its impact on prediction of adaptive maintenance using MLR for OO systems. The validation has identified that several metrics can be profitably employed for the prediction of software maintainability. Misra [23] used linear regression in his study which was

based on intuitive and experimental analyses using twenty design and code measures to obtain their indications on software maintainability. In the last decade some machine learning algorithms have also been proposed, evaluated and verified that they can predict maintainability more accurately and precisely. Thwin and Quah [10] used Artificial Neural Network (ANN), Koten and Gray [11] applied Bayesian Belief Network (BBN), Elish and Elish [12] applied Tree Nets in maintainability prediction modeling for OO systems. Kaur et al. [14] have verified the use of soft computing approaches for maintainability prediction to achieve more accuracy. Recently many nature inspired algorithm are successfully applied such as evolutionary programming for open source software systems by Banker et al. [24], ant colony optimization used by Sun and Wang [25] for optimizing preventive maintenance and genetic algorithms by Vivanco et al. [26].

3 Proposed Metrics

It's important to give equal attention to the database accesses with the enhancement in data base usage now days. With the increase in the use of mobile and mobile based applications, data that once might have been accessed a couple of times a week now might be accessed multiple times per hour. As the software systems heavily use data bases; hence we observed that C&K metric suite would not be adequate as it does not capture the database handling aspects of the applications. We proposed two more metrics namely SCCR and NODBC as presented in Table 1, to remove these deficiencies and we claim that two proposed metrics carries more impact on software maintainability in database intensive applications.

NODBC is measured by counting the number of times database connections were made using the function call 'Open()'. To count the SCCR, ratio of the numbers of field in the schema to the number of comment lines was considered. Authors are of the strong opinion that understandability of the schema of database is equally important in maintaining any application.

4 Research Background

Independent and dependent variable: To validate the effectiveness of proposed metric suite, 10 independent variables have been considered as compiled in Table 2. C&K metric suite is used to measure OO characteristics and MI as well as CC were used to capture the structural complexity of the code. Inspired by the results of Malhotra and Chug [27], NODBC and SCCR also added to measure the data base handling aspect.

Empirical data collection: Five proprietary systems were considered as presented in Table 3. To calculate the values of all independent variables, following

Table 1 Proposed metrics

Metric name	Description
Scheme complexity to comment ratio (SCCR)	It calculates the ratio of number of comments lines to the number of field in the schema of data base
Number of data base connections (NODBC)	Number of data base connection is a measure to count number of times database connections were made

Table 2 Set of independent variables

Metric name	Description
Weighted methods per class (WMC)	It is the sum of McCabes's cyclomatic complexities of all local methods in a class
Depth of inheritance tree (DIT)	It measures the depth of a class in terms of the distance from root class; the value of DIT for root class is zero
Number of children(NOC)	It measures the number of child classes of a class as it counts number of immediate sub classes
Coupling between object (CBO)	It count the number of other classes to which the given class is coupled
Response for a class (RFC)	It counts the number of functions executed in response to the received message
Lack of cohesion of methods (LCOM)	It counts the number of disjoint sets of local methods
Scheme complexity to comment ratio(SCCR)	It calculates the ratio of number of comments lines to number of field in the schema of data base
Number of data base connections (NODBC)	Number of data base connection is a measure to count number of times database connections were made
Maintainability index (MI)	Calculates an index value between 0 and 100 that represents the relative ease of maintaining the code
Cyclomatic complexity (CC)	Measures the structural complexity of the code

Table 3 Brief description of the proprietary software systems used in the empirical study

System name	Data points	Description
File letter monitoring (FLM system)	233	Customize software to handle the movement of files and documents in an office
EASY system	292	Web portal for an educational institute to provide study material and online evaluations
Student management system (SMS system)	129	Maintains the record of students and teacher for private educational institute
Inventory management system (IM system)	96	Maintains inventory of the company at different branch offices in different cities
Angel bill printing (ABP system)	114	Maintains fully editable items list by client itself with generation of a common bill format

strategy is used. Five metrics namely MI, CC, DIT, CBO and NOC were retrieved from the Visual Studio wherein the metrics mentioned were calculated from the intermediate language code generated while compilation. Three metrics namely

Table 4 Descriptive statistics of FLM system and EASY system

Metric	Max	Min	Mean	Med	SD	Max	Min	Mean	Med	SD
WMC	16	1	6.276	5	4.97	23	1	10.5	9.5	8.57
DIT	7	1	4.379	5	1.32	5	1	3.6	4	2.50
NOC	7	0	3.1	3	1.67	8	0	4.23	3	2.91
CBO	50	3	26.14	30	13.85	54	0	33.5	38.5	21.58
RFC	67	12	25.16	18	7.89	78	21	37.73	27	4.89
LCOM	0	0	0	0	0	0	0	0	0	0
SCCR	5	2	3.276	3	2.97	7	3	4.57	5	5.57
NODBC	12	0	2.483	0	3.53	7	0	2.79	0.5	3.43
MI	91	40	61.14	56	18.04	94	43	64.1	56.5	17.91
CC	29	1	19.31	16	13.76	22	1	20.6	19	14.26
Change	95	5	41.98	67	45.67	87	9	52.52	63	43.23

Table 5 Descriptive statistics of SMS system and IMS system

Metric	Max	Min	Mean	Med	SD	Max	Min	Mean	Med	SD
WMC	29	2	16.63	17.5	9.17	12	0	3.147	3	2.57
DIT	6	1	3.25	4	2.12	5	4	4.029	4	0.17
NOC	11	0	4.85	4	2.67	7	0	2.81	3	1.91
CBO	59	3	45.38	52.5	18.66	30	2	13	13.5	8.09
RFC	83	19	37.09	31	5.87	43	18	21.09	27	5.07
LCOM	0	0	0	0	0	3	0	0.147	0	0.55
SCCR	6	2	4.625	16.5	9.17	12	0	3.147	3	2.57
NODBC	6	0	3.89	3	2.50	5	0	2.118	1	3.85
MI	81	49	55.25	52	10.56	100	48	71.79	67	17.84
CC	27	1	21.5	19.5	19.61	13	2	10.79	7	12.38
Change	79	13	67.89	47	32.43	213	18	79.87	103	67.93

WMC, LCOM and RFC were calculated with the help of CCCC tool [28]. Remaining two metrics as proposed in this study SCCR and NODBC were collected through tool we created in the first phase of our research plan. We observed the software over a period of 3 years since it has been delivered. Original as well as modified versions were compared manually to count the CHANGE i.e. dependent variable. Any line of source code added or deleted is counted as one whereas modification counted as two changes. The value of change for each class was compiled and combined with respective values of independent variables to generate the data points. Same methodology was adopted in Zhou et al. [8] and Malhotra et al. [29]. We found 233, 292, 129, 96 and 114 data points for FLM, EASY, SMS, IMS and ABP System respectively.

Descriptive statistics such as Max, Min, Mean, and Median (Med) and Std Dev(SD) were calculated for FLM and EASY systems and presented in Table 4, SMS and IMS system presented in Table 5 whereas ABP system presented in Table 6. From the table it can be observed that the Max value of LCOM for FLM, EASY, SMS, IMS and ABP are 0, 0, 0, 3 and 6 respectively which represents that

Table 6 Descriptive statistics of ABP system

Metric	Max	Min	Mean	Med	SD
WMC	11	1	2.483	2	1.84
DIT	6	3	4.017	4	0.13
NOC	9	0	5.25	5	1.09
CBO	29	4	14.93	17	8.56
RFC	49	21	26.83	31	9.89
LCOM	6	0	0.155	0	0.81
SCCR	11	1	2.483	2	1.84
NODBC	8	0	4.931	1	1.04
MI	100	40	69.5	61	21.03
CC	14	2	10.33	8.5	8.88
Change	189	19	91.23	78	45.63

Table 7 Pearson correlation coefficient at 0.01 level of significance (two tailed)

Metrics	FLM change	EASY change	SMS change	IMS change	ABP change
WMC	**0.73**	**0.66**	**0.54**	**0.61**	**0.59**
DIT	−0.38	−0.42	−0.36	−0.42	−0.44
NOC	0.29	0.48	0.33	0.41	0.45
CBO	0.46	**0.61**	0.49	**0.58**	0.51
RFC	**0.64**	0.49	**0.50**	**0.51**	0.47
LCOM	0.48	0.42	0.41	**0.68**	**0.71**
SCCR	**0.56**	**0.55**	**0.66**	**0.69**	**0.73**
NODBC	**0.71**	**0.65**	**0.58**	**0.79**	**0.81**
MI	**0.61**	0.49	0.36	0.47	**0.58**
CC	**0.59**	**0.62**	0.39	0.41	**0.55**

classes are quite cohesive in first three applications. Values of DIT for FLM, EASY, SMS, IMS and ABP are 7, 5, 6, 5 and 6 which represents that inheritance is properly exploited in all systems. SCCR is medium in FLM, EASY and SMS and High in IMS and ABP which means IMS and ABP would be easier to understand in maintenance phase. A value of NODBC is more than 8 in FLM and ABP systems and less than 7 in EASY, SMS and IMS systems.

Correlation Analysis: Correlation Analysis provides important information about the interdependence between two variables. We calculated the Pearson's correlation coefficient represented as 'r' to measures the linear relationship between independent variables versus change and presented in Table 7. Value of 'r' represents the amount of correlation exists between the two variables and lies between +1 to −1. Values in the range of ±0.5–1 represent high correlation; ±0.3–0.5 represents medium correlation whereas less than ±0.3 represents very low correlation. In the Table 7, all entries above 50 % are marked as bold. It is inferred that NODBC metric as well as SCCR metric is significantly related to change metric for all the systems. The value of 'r' for new proposed metric is quite competitive as compared to other metrics. For IMS and ABP systems, more than

75 % correlation was observed whereas for FLM, EASY and SMS systems it was in the range of 58–75 % which is quite significant. SCCR is also found to be significantly correlated with change metric for all systems. When compared with other metrics it was found that although DIT is comparatively less correlated with the change however MI and CC are reasonably well correlated. Among the C&K metric suite, WMC is found to be most significantly related as for all systems, value of 'r' is found to be more than 54 % for all systems. RFC is significantly correlated with change for FLM, SMS and IMS systems. CBO found to be significantly correlated with change in EASY and IMS systems.

5 Research Methodology

In this section, we explain the various Machine Leaning (ML) methods used for making the prediction models as well as to ascertain the relationship of design metrics with maintainability. Recent research activities carried by authors [7, 15, 27, 29] have revealed that ANN is very powerful in classifying and recognizing the data patterns, so they are well suited for prediction problems as in such cases although the required knowledge is difficult to specify but enough data for observations are available to learn. They are originally developed to mimic basic biological neural systems particularly the neurons present in the human brain. Four different versions of ANN models have been selected in the current study as mentioned below.

(a) Back Propagation Network (BPN): Although BPN is originally invented by Hu [30] in 1964 however it came into use only in 1986 by Rumelhart et al. [31] when it was used as supervised learning technique. Training data in BPN consists of pair of vector (input vector and target vector). During the training process, an input vector is presented to the network for the learning process. Output vector is generated from these learning and compared with the actual target vector. If there is any difference in the values, the weights of the network are re-adjusted to reduce this error and the process is repeated until the desired output is produced.

(b) Kohonen Network (KN): Proposed by Kohonen [32], KN is best known as self organizing networks as they learn to create maps of the input space in a self-organizing way. Although, KN is invented to provide a way of representing multidimensional data in much lower dimensional spaces, a network is created that learn the information such that any topological relationships within the training set are maintained without supervision.

(c) Feed Forward Neural Network (FFNN): In FFNN [33, 34], information moves in only one direction i.e. forward from input nodes to output nodes through hidden nodes and there are no loops in the network. The number of hidden neuron selected as 10 for the sample data collected from these five real life applications.

(d) General Regression Neural Networks (GRNN): Proposed by Specht [35], it is very powerful network as it needs only a fraction of the training samples during learning process and finishes the learning process in single pass. Due to the highly parallel structure, it performs well even in case of noisy and sparse data and the over fitting problem does not arise as neither do they set the training parameters during the commencement of learning process, nor they define the momentum. Once the network finished the training process, only smoothing factor is applied to determine how tightly the network matches its prediction [10].

6 Results and Discussion

Ten independent and one dependent variable were selected in this study. Total 864 classes were collected and combined with respective changes made in each class. Univariate and Multivariate analysis was performed to find the significance of each metric individually and cumulatively on change.

Univariate Analysis using linear regression was performed to find the individual effect of NODBC and SCCR on change using SPSS and the results are presented in Table 8. Four columns represent estimated coefficient, standard error, the t-ratio and p-value. The value of Sig (p-value) represents amount of significance of these metrics on change. As evident from the outcome, both variables received the p-value as 0.000 which means they are significantly correlated with change.

Multivariate Linear Regression (MLR) was also performed using stepwise linear regression model in order to identify the most significant metrics for each system. MLR is the most commonly used technique for fitting a linear equation on observed data [8]. There are three methods used for identifying and picking the subset of important metrics from the set of independent variables i.e. forward selection, backward selection and stepwise selection. In this study, stepwise selection method is used as it guarantees to provide optimum and most significant subset of independent variables. At each step either the certain variables are added or deleted to identify the final most optimized regression model. Unstandardized Coefficient, Std Error, t-ratio and p-value (sig) to three decimal places are presented in Table 9.

Results show that two proposed metrics were found to be statistically significant for all systems as almost all p-value are less than .050. Unstandardized Coefficients represents the value when the dependent and independent (predictor) variables were all transformed to standard scores before running the regression and used to compare the relative strength of the various predictors. NODBC has the largest coefficient and one standard deviation increase in NODBC leads to a 0.915 decrease in change for IMS system. SCCR is also found to be quite competitive as one standard deviation increase in SCCR in turn leads to 0.858 standard deviation

Table 8 Results of univariate analysis

Metric	Unstandardized coefficient	Std. error	t-ratio	Sig (p-value)
NODBC	0.307	0.020	5.101	0.000
SCCR	0.245	0.046	2.826	0.000

Table 9 Results of multivariate analysis

Software system	Most significant metrics identified	Unstandardized coefficient	Std. error	t-ratio	Sig (p-value)
FLM system	Intercept	3.779	1.054	3.586	0.000
	WMC	0.009	0.004	2.532	0.012
	MI	0.055	0.067	2.489	0.014
	SCCR	0.485	0.015	2.203	0.000
EASY system	Intercept	0.542	0.191	2.838	0.005
	WMC	0.023	0.226	1.882	0.018
	SCCR	0.378	0.773	2.563	0.000
	NODBC	0.584	0.798	2.066	0.000
SMS system	Intercept	0.697	0.854	1.806	0.000
	NODBC	0.860	0.211	0.055	0.002
	MI	0.707	2.876	0.013	0.004
	SCCR	0.858	0.463	0.019	0.001
IMS system	Intercept	0.912	0.687	0.258	0.000
	SCCR	0.345	0.605	0.069	0.004
	NODBC	0.915	2.463	0.150	0.002
	RFC	0.301	4.501	0.663	0.000
ABP system	Intercept	0.032	1.268	0.757	0.003
	WMC	0.817	3.412	0.681	0.010
	NODBC	0.476	3.406	0.146	0.004
	MI	0.817	3.412	0.681	0.010

increase in change for SMS system. Apart from two reported metrics WMC and MI were also found to be most significant predictor of change.

Maintainability Prediction: Two types of prediction models were constructed for each system. **Model-1** is constructed using metrics suite presented by C&K [5] and **Model-2** is constructed by adding four more metrics MI, CC, NODBC and SCCR to the existing C&K metrics suite resulting in the set of 10 metrics in all. MLR, BPNN, KN, FFNN and GRNN were employed for software maintainability prediction by dividing the data into three parts i.e. 70 % for training and 30 % for testing as it is the commonly accepted proportion used by many practitioners [6–15]. Three prediction accuracy measures proposed by Kitchenham et al. [36] as presented in Table 10 are used to compare the performance of Model-1 and Model-2. Detailed method for their calculations are available in Malhotra et al. [15].

Results are presented in Table 11 where three rows for each software system represent the values of accuracy measures when MLR as well as ML models were applied with metric suite Model-1 (M-1) and metric suite Model-2 (M-2). For

Table 10 Prediction accuracy comparison proposed by Kitchenham et al. [36]

Name	Formula	Definition
MRE (Magnitude of relative error)	$\frac{\lvert Actual - Predicted\ Value \rvert}{Actual\ Value}$	Normalized measure of the discrepancy between actual and predicted value
MMRE (Mean magnitude of relative error)	$\sum_{i=1}^{N} MRE_i$	Average relative discrepancy
Pred(q)	$Pred(q) = \frac{K}{N}$	What proportion of the predicted values have MRE less than or equal to specified value

Table 11 Prediction accuracies of model-1(M-1) and model-2 (M-2) for all data sets

Software system	Accuracy measures	MLR model		BPNN model		KN model		FFNN model		GRNN model	
		M-1	M-2	M-1	M-2	M-1	M-2	M-1	M-2	M-1	M-2
FLM system	Max MRE	2.14	1.53	1.98	1.20	1.87	1.09	1.332	0.98	1.78	1.11
	MMRE	0.51	0.48	0.49	0.47	0.45	0.42	0.41	0.39	0.48	0.41
	Pred(0.25)	0.64	0.78	0.57	0.69	0.68	0.78	0.66	0.71	0.68	0.76
Easy system	Max MRE	2.09	1.82	1.24	1.65	0.99	0.98	1.090	1.02	1.47	1.32
	MMRE	0.66	0.57	0.51	0.46	0.46	0.36	0.43	0.37	0.49	0.42
	Pred(0.25)	0.58	0.63	0.59	0.63	0.69	0.74	0.70	0.77	0.68	0.69
SMS system	Max MRE	1.98	1.47	1.30	1.22	1.11	0.97	2.786	1.70	1.98	1.8
	MMRE	0.68	0.56	0.44	0.40	0.43	0.41	0.46	0.38	0.42	0.39
	Pred(0.25)	0.60	0.66	0.52	0.69	0.63	0.77	0.52	0.69	0.60	0.71
IMS system	Max MRE	2.34	1.71	1.52	1.43	2.33	1.10	1.803	1.22	1.88	1.63
	MMRE	0.71	0.59	0.40	0.35	0.43	0.32	0.39	0.30	0.38	0.29
	Pred(0.25)	0.54	0.63	0.59	0.71	0.68	0.70	0.59	0.63	0.58	0.69
ABP system	Max MRE	1.78	1.37	1.29	1.33	1.89	0.33	2.092	1.66	1.67	1.45
	MMRE	0.49	0.41	0.37	0.29	0.43	0.32	0.48	0.37	0.58	0.40
	Pred(0.25)	0.69	0.76	0.58	0.67	0.62	0.72	0.67	0.66	0.71	0.74

example first three rows belong to the results received using FLM system as MLR, BPNN, KN, FFNN and GRNN models were applied for each prediction algorithms with two different data sets i.e. Model-1 (M-1) and Model-2 (M-2).

From the results it is quite evident that overall improvement in the prediction accuracy is observed with new proposed metric suite for all systems. To further analyze the results we further sorted the systems in ascending order on the values of NODBC and SCCR. We observed that more improvement in prediction accuracy was achieved for those systems which have high values of NODBC and SCCR. ABP system has maximum SCCR and NODBC as compared to other systems. For ABP system maximum improvement in prediction accuracy is observed i.e. 23 % in the for MMRE whereas other systems such as FLM, EASY, SMS and IMS observed 7, 14, 11 and 19 % improvement in MMRE respectively. MaxMRE was improved by 39, 1, 21, 28 and 29 % for FLM, EASY, SMS, IMS and ABP System respectively. Lowest improvement for Easy systems was noticed

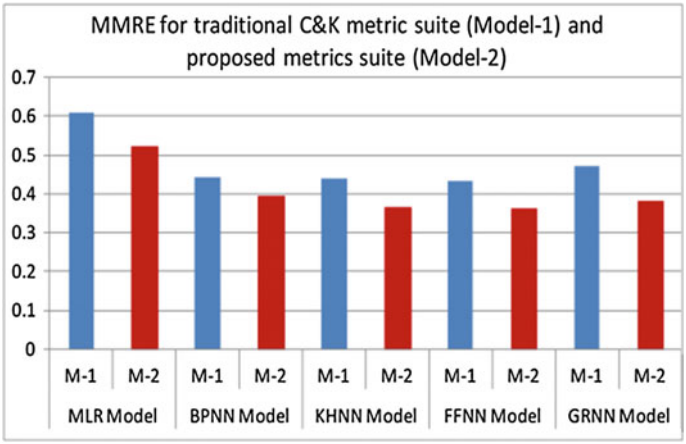

Fig. 1 MMRE for Model-1 and Model-2

which also has lowest SCCR as well as NODBC among all systems. Prediction accuracies achieved by all models were also compared and observed that the performance of ML models is better than MLR in general. When we compared the MMRE values for Model-2, it is found to be 0.94, 0.82, 0.66, 0.79 and 0.86 for MLR, BPNN, KN, FFNN and GRNN respectively. That means KN performance is best among all ML models. Graphs were also plotted to observe improvement in prediction accuracies from Model-1 to Model-2 w.r.t. MMRE and Pred(0.25) in Figs. 1, 2 respectively. It is quite evident from Fig. 1 that MMRE was significantly reduced from Model-1 to Model2 for all prediction techniques. Figure 2 represents the comparison of prediction accuracies achieved at 25 %. It is quite visible from the graph that pred(0.25) is improved from Model-1 to Model-2 for all techniques.

7 Threats to Validity

Whenever any empirical data is collected from proprietary software system, it has got few specific characteristics and their generalization always carries few threats to its validity. Also in this study, OO characteristics were measured using internal quality metrics suite proposed by C&K. However, software maintainability also depends upon external quality attributes such as competency of developers, familiarity with the code etc. They were intentionally avoided due to the subjectivity involved in their measurement. We also cannot assure if the proposed metrics suite is universally applicable for different programming languages and environment. In order to capture the cause-effect relationship between particular metric and maintainability, we need to perform controlled experiment where one metric is kept constant and others varied. This threat also exists in our study as carrying such experiments is extremely difficult.

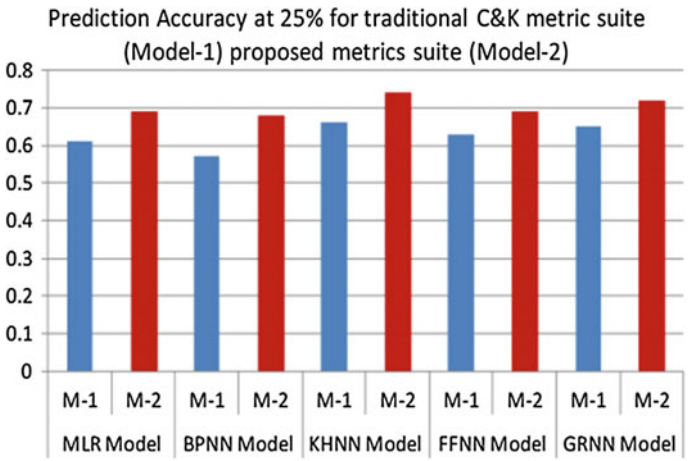

Fig. 2 Pred(0.25) for Model-1 and Model-2

8 Conclusion and Future Work

The goal of our research was to empirically examine the effectiveness of new proposed metric suite for predicting software maintainability for data intensive applications as it's important to give equal attention to the database accesses with the increase in data as well as the number of times data get accessed. We employed MLR, BPNN, FFNN, KN, and GRNN techniques for making software maintainability prediction model. Observing five proprietary software over a period of 3 years, we analyzed the performance of proposed metric suite using prediction accuracy measures such as MRE, MMRE and pred(0.25). Four more metrics were added (MI and CC for measuring the structural complexity and NODBC and SCCR for measuring the database aspect) to the traditional C&K metrics suite. Main results of the current study are summarized as follows:

- The predicted results indicate that proposed metric suite is significant indicator of software maintainability, as improvements in all five datasets were observed when four more metrics added to the C&K metric suite.
- The results received from pearson's correlation coefficient safely suggest that proposed metrics were significantly correlated with change.
- The predicted results indicate that we can use KN in building maintainability prediction models in data intensive applications.
- Multivariate analysis using stepwise linear regression identified NODBC and SCCR as good indicator of software maintainability in data intensive applications.

Result of this study helps practitioners in using new metric suite for developing maintainability prediction models. The results help us in identification of those classes which require big share of maintenance resources and the limited resources

can be planned accordingly. The results of our study are valid for medium systems developed in C#. In future, we plan to replicate our studies on data sets having different characteristics such as datasets with different programming languages and environments.

References

1. IEEE Standard 1219–1993. IEEE Standard for Software Maintenance. INSPEC Accession Number: 4493167 doi: 10.1109/IEEESTD.1993.115570 June, 1993
2. Software Engineering Standards Committee of the IEEE Computer Society, IEEE Std. 828–1998 IEEE Standard for Software Configuration Management Plans, http://standards.ieee.org/findstds/standard/828–1998.html
3. K.K. Aggarwal, Y. Singh, A. Kaur, R. Malhotra, Analysis of object-oriented metrics, in *International Workshop on Software Measurement (IWSM)*, 2005
4. R. Bandi, Predicting maintenance performance using object-oriented design complexity metrics. IEEE Trans. Softw. Eng. **29**(1), 77–87 (2003)
5. S. Chidamber, C. Kemerer, A metrics suite for object oriented design. IEEE Trans. Softw. Eng. **20**(6), 476–493 (1994)
6. W. Li, S. Henry, Object-oriented metrics that predict maintainability. J. Syst. Softw. **23**, 111–122 (1993)
7. K.K. Aggarwal, Y. Singh, A. Kaur, R. Malhotra, Application of artificial neural network for predicting maintainability using object oriented metrics. Proc. World Acad. Sci. Eng. Technol. **15**, 285–289 (2006)
8. Y. Zhou, H. Leung, Predicting object-oriented software maintainability using multivariate adaptive regression splines. J. Syst. Softw. **80**(8), 1349–1361 (2007)
9. M. Dagpinar, J.H. Jahnke, Predicting maintainability with object-oriented metrics: an empirical comparison, in *Proceedings of the 10th Working Conference on Reverse Engineering (WCRE '03)*, IEEE Computer Society, Washington, 2003
10. M. Thwin, T. Quah, Application of neural networks for software quality prediction using object oriented metrics. J. Syst. Softw. **76**(2), 147–156 (2005)
11. C.V. Koten, A.R. Gray, An application of Bayesian network for predicting object-oriented software maintainability. Inf. Softw. Technol. **48**(1), 59–67 (2006)
12. M.O. Elish, K.O. Elish, Application of TreeNet in predicting object-oriented software maintainability: a comparative study, in *Proceedings of European Conference on Software Maintenance and Reengineering*, 2009
13. C. Jin, J.A. Liu, Applications of support vector machine and unsupervised learning for predicting maintainability using object-oriented metrics. in *Proceedings of the 2nd International Conference on Multi Media and Information Technology*, 2010
14. A. Kaur, K. Kaur, R. Malhotra, Soft computing approaches for prediction of software maintenance effort. Int. J. Comput. Appl. **1**(16), 975–988
15. R. Malhotra, A. Chug, Software maintainability prediction using machine learning algorithms. Softw. Eng. Int. J. **2**(2), 19–36 (2012)
16. W. Li, Another metric suite for object-oriented programming. J. Syst. Softw. **44**(2), 155–162 (1998). doi:10.1016/O164-1212(98)10052-3
17. P. Oman, J. Hagemeister, Metrics for assessing a software system's maintainability, conference on software maintenance (IEEE Computer Society Press, Los Alamitos, 1992), pp. 337–344
18. P. Oman, J. Hagemeister, Construction and testing of polynomials predicting software maintainability. J. Syst. Softw. **24**, 251–266 (1994)
19. T. McCabe, A complexity measure. IEEE Trans. Softw. Eng. **SE-2**(4), 308–320 (1976)

20. M. Jorgensen, Experience with the accuracy of software maintenance task effort prediction models, IEEE Transc. Softw. Eng. **21**(8), 674–681 (1995)
21. S. Muthanna, K. Kontogiannis, B. Ponnambalam, A. Stacey, Maintainability model for industrial software system using design level metrics, in *Proceedings of 7th Working Conference on Reverse Engineering*, 2000, pp. 248–256
22. F. Fioravanti, P. Nesi, Estimation and prediction metrics for adaptive maintenance effort of object-oriented system. IEEE Trans. Softw. Eng. **27**(12), 1062–84 (2001)
23. S.C. Misra, Modeling design/coding factors that drive maintainability of software systems. Softw. Qual. J. **13**(3), 297–320 (2005)
24. A.D. Banker, A.B. Sultan, H. Zulzalil, J. Din, Applying evolution programming search based software engineering (SBSE), in *Proceedings of Selecting the Best Open Source Maintainability Metrics, International Symposium, ISCAIE, 2012*
25. P. Sun, A. Wang, Application of ant colony optimization in preventive software maintenance policy, in *Proceedings of IEEE international Conference on Information Science and Technology*, China, Mar 2012
26. R. Vivanco, N. Pizzi, Finding effective software metrics to classify maintainability using a parallel genetic algorithm. Genetic Evol. Comput. (Lecture Notes in Computer Science) **30**(13), 1388–1399 (2004)
27. R. Malhotra, A. Chug, An empirical study to redefine the relationship between software design metrics and maintainability in high data intensive applications, in *Proceedings of the World Congress on Engineering and Computer Science 2013, WCECS 2013*. Lecture Notes in Engineering and Computer Science, San Francisco, 23–25 Oct, 2013, pp. 61–66
28. C and C++ Code Counter to Compute OO Metrics and the McCabe Cyclomatic Complexity Metrics from Source Code. http://cccc.sourceforge.net/
29. R. Malhotra, A. Chug, An empirical validation of group method of data handling on software maintainability prediction using object oriented systems, in *Proceedings of International Conference on Quality Reliability InfoCom Technology and Industrial Technology, ICQRITM 2012*, New Delhi, pp. 49–57
30. M.C.J. Hu, Application of the adaline system to weather forecasting. Master thesis, technical report 6775-i, Stanford Electronics Laboratories, 1964
31. D.E. Rumelhart, G.E. Hinton, R.J. Williams, *Learning Internal Presentation by Back-Propagating Errors, the PDP Research Group, Parallel Distributing Processing: Exploration in the Microstructure of Cognition* (MIT Press, MA, 1994)
32. T. Kohonen, *Self-Organization and Associative Memory* (Springer, Berlin, 1989)
33. A.E. Bryson, Y.C. Ho, *Applied Optimal Control: Optimization, Estimation, and Control* (Blaisdell Publishing Company, New York, 1969), p. 481
34. S. Russell, P. Norvig, *Artificial Intelligence: A Modern Approach*, 2nd edn. (Prentice Hall, Upper Saddle River, 2003)
35. D.F. Specht, A general regression neural network. IEEE Trans. Neural Netw. **2**(6), 568–576 (1991)
36. B.A. Kitchenham, L.M. Pickard, S.G. MacDonell, M.J. Shepperd, What accuracy statistics really measure. IEE Proc. Softw. **148**(3), 81–85 (2001)