Using Source Code Metrics and Multivariate Adaptive Regression Splines to Predict Maintainability of Service Oriented Software

Lov Kumar[1]
Dept. CS&E
NIT Rourkela, India
lovkumar505@gmail.com

Santanu Ku. Rath[2]
Dept. CS&E
NIT Rourkela, India
skrath@nitrkl.ac.in

Ashish Sureka[3]
ABB (India)
ashish.sureka@in.abb.com

***Abstract*—Prediction of maintainability parameter for Object-Oriented software using source code metrics is an area that has attracted attention of several researchers. However, maintainability prediction of Service-Oriented software is a relatively an unexplored area. In this work, we conduct a experiment on maintainability prediction of eBay web services using source code metrics. We have considered eleven different types of source code metrics as input to develop a maintainability prediction model using multivariate adaptive regression splines (MARS) method. The performance of the maintainability prediction model is evaluated and compared with multivariate linear regression (MLR), and support vector machine (SVM). Since the performance of the classifiers depends on the source code metrics, they are used as input of the classifier. So, eight different types of feature selection techniques have been implemented to reduce dimension and remove irrelevant features. The experiment results ravel that the maintainability prediction model developed using MARS method achieved better performance as compared to MLR and SVM methods. Experiment results also demonstrates that the model developed by considering selected set of source code metrics by feature selection technique as input achieves better results as compared to considering all source code metrics.**

***Keywords*—SOA, MARS, MLR, Maintainability, Feature Selection Technique.**

I. Introduction

Service-Oriented Computing (SOC) paradigm is the dominant development platform for software systems today. In this development platform, developer assembles loosely coupled pieces of software (service), and construct the distributed system. With the increasing complexity and size of SOC paradigm, one of major objectives of software engineering is to control the development of software process by controlling costs and schedules, as well as the quality of the software products. Software maintenance is the most expensive phase for any software product over its lifetime [1]. One most effective way to control the maintenance costs is to utilize software metrics during the development phase [2]. The ISO/IEC 9126 [3] standard defines software maintainability as the capability of the software product to be modified, including adaptation or improvements, corrections of the software to changes in environment, requirements and functional specifications. Software maintainability can be further subdivided into four sub-characteristics such as analysability, changeability, stability, and testability. In this work, we focus on one sub-characteristics of maintainability i.e., changeability of Service-Oriented Computing paradigm.

Web Services are observed to be described using a WSDL (Web Service Description Language) document. The literature is quite rich in research articles in the area of predicting maintainability of object oriented software (both close and open source software) using source code metrics [4] [5] [6]. However, the problem of maintainability prediction for web service interfaces and WSDL documents is relatively unexplored. Our research aims to investigate the usefulness of metrics for the source code implementing a web service for maintainability prediction. The research work further looks into the aspect of usefulness of source code metrics for developing a model to predict the maintainability of web service. In this process, we examine the application of multivariate adaptive regression splines (MARS) method to build estimators based on source code metrics as predictor variables and maintainability as target variable.

MARS method uses the series of piecewise regression splines for adapting the unknown functional form. MARS concept is based on divide-and-conquer algorithm, which separated the data into various regions. Each region has its own regression equation. The application of MARS method in various domains have been mentioned in articles[1] [7]. Three different performance parameters such as accuracy, precision, and recall have been considered for evaluation of MARS models. In this paper, we also compare the performance of MARS model with performance of the model developed using multivariate linear regression models (MLR), and support vector machine (SVM) methods.

The source code metrics are considered as predictor variables to develop a maintainability prediction models. This shows that the performance of maintainability prediction model also depends on the predictor variables i.e., source code metrics. So in this work, eight different types of features selection techniques have been considered to select the right set of source code metrics. These selected set of metrics may improve the performance of maintainability prediction model and also reduce the number of misclassification errors. Each feature selection technique consider all available features and derives a right set of features. Various applications of feature selection techniques in different domains have been reported [8][9].

Two different class of feature selection techniques such as feature ranking (FR) and feature subset selection (FSS) techniques are used to remove irrelevant features and select best set of features. In FR technique, some performance parameters are used to rank the features and some features are selected which are suitable for a given project. In case of FSS technique, best subset of features are searched which collectively have good predictive capability. In this experiment, we have considered four types of FR techniques such as Gain Ratio (GR) Feature Evaluation [10], Chi Squared (CH) test [11], OneR Feature Evaluation [10] and Information Gain (IG) Feature Evaluation [10] and four types of FSS techniques such as Classifier Subset Evaluation [12], Correlation based Feature Selection, Filtered Subset Evaluation [13], Rough Set Analysis (RSA) [14] to compute optimal set of source code metrics for improving the performance of web-service maintainability prediction model. The performance of selected sets of metrics using feature selection techniques have been evaluated using three different classification techniques such as MLR, MARS, and SVM.

Research Contributions: Our contributions in this conference include :

- Development of Data Collection and Preparation Process.
- Development of a maintainability prediction model for service oriented application using MARS and source code metrics.
- Selection of right set of source code metrics to improve the performance of maintainability prediction model.

II. Related work

while referencing to the development process using service oriented architecture (SOA), Grady Booch in an interview conducted by Info World (2004) stated that "you start with web services and you start with good solid object-oriented architectures". He justified this by stating that "the fundamentals of engineering like good abstractions, good separation of concerns never go out of style", but "there are real opportunities to raise the level of abstraction again". Many research works have investigated the effects of adopting the Object-Oriented (OO) approach on software quality and observed that OO paradigm has a profound impact on software quality attributes. We hypothesize that the Quality of OO software can be best estimated using several source code metrics [15] [4] [16]. In this section, we highlight some previous work on application of source code metrics for improving quality of service-oriented applications.

Bingu Shim et al. defined five different quality parameters i.e., discoverability, effectiveness, flexibility, reusability, and understandability for service oriented applications [17]. They have observed that, these quality attributes are depends on various number of design properties, i.e., consumability, cohesion, coupling, complexity, size, different types of granularity etc.. In this work, they have derived some metrics and compute the relation between design metrics.

Cristian Mateos et al. analyzed the available approaches to remove undesirable anti-patterns using code-first [18]. They worked on the hypothesis that object-oriented source code metrics avoid the occurrence of anti-pattern. Mateos et al. considered twelve different types of source code metrics such as: AMC, CAM, CBO, WMC, LOC, LCOM, LCOM3, RFC, EPM, VTC, ATC, DAM to find the anti-pattern occurrence at the WSDL level.

Mikhail Perepletchikov et al. proposed cohesion and coupling metrics for predicting maintainability parameters of Service-Oriented Software [19] [20]. In this work, they redefined the cohesion and coupling metrics for service-oriented software and mentioned the application of these metrics on system maintainability in terms of analysability, changeability, stability, and testability.

From literature, we observed that the OO source code metrics are being used for quality indicators of SOA system [21] [18] [22]. The limitations of these studies may be inferred that the authors focus on refining and proposing new source code metrics for improving the quality of service oriented applications. Hence in this work, our objective is to develop a model using source code metrics as input for predicting one quality attribute i.e., maintainability of SOA. This model is developed using multivariate adaptive regression splines (MARS).

Since, source code metrics are considered as input of the models, so the performance of models also depends on subset of source code metrics. We have considered eight different types of feature selection techniques to select suitable set of features which help to predict maintainability with higher accuracy.

III. Experimental Dataset and Setup

A. Source Code Metrics

Based on our analysis of previous work done in this direction, we observe that authors have considered different set of source code metrics for different application i.e., fault prediction, effort estimation, reliability prediction, maintainability prediction etc.. The most commonly used source code metrics are: Li and Henry [4], Abreu MOOD metrics suite [23], McCabe [24], Halstead [25], Briand *et al.* [26], CK metrics [15] suite, Lorenz and Kidd [27] etc.. In this work, eleven different source code metrics i.e., Depth of inheritance tree (DIT), Weighted method per class (WMC), Coupling between object (CBO), Number of children (NOC), Number of methods (NOM), Lack of cohesion among methods (LCOM), Response for class (RFC), Data abstraction coupling (DAC), SIZE2, SIZE1, McCabe Cyclomatic complexity (MVG) are computed to develop a predictive model for estimating maintainability of web services. These source code metrics are computed using tool such as CKJM and LOC metrics. These metrics include the popular metrics suite proposed by Chidamber and Kemerer [28], and Li and Henry [4]. The explanation of these source code metrics are given in [4].

B. Data Collection and Preparation Process

Figure 1 displays the multi-step process of data collection and preparation. Five different versions of eBay web services are obtained from a publicly available source[1], with an intention that our experiments can be replicated and used for benchmarking and comparison.

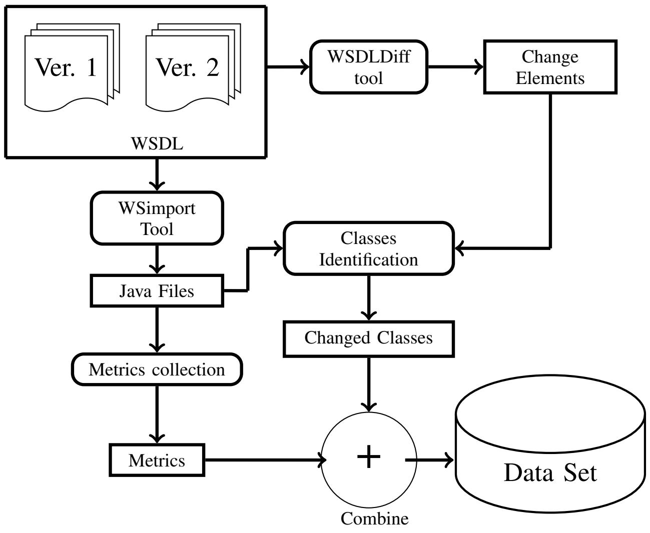

Fig. 1: Data Collection and Preparation Process

The following steps are followed in our proposed experiment.

Step 1: *WSDLDiff Tool:* In this paper, WSDLDiff tool has been used for comparing subsequent versions of WSDL interfaces ([29]) and finding the name (Operation, Message etc.) and type (addition, move, removed, update etc.) of the elements affected by the changes. These elements are called as 'changed elements (Operation, Message etc.)'.

Step 2:*WSimport Tool:* Some tools such as Apache CXF, Soap UI, Wsimport etc. are available in literature [19] [20] to Java files from WSDL document. In this work, we have considered Wsimport tool to parse WSDL document of web-service and generate Java files.

Step 3: *Metrics Collection:* Metrics values of source code are computed using tools such as CKJM and LOC metrics.

Step 4: *Class Identification:* The Java classes which contain the changed elements (Operation, Message etc.) are termed as changed classes. These types of classes containing changed elements are identified.

Step 5: *Data Set:* The change statistics from Step 4 and metrics from Step 3 are combined to generate data set for further processing.

C. Case Study

From literature survey, we observed that the most of the authors only use two versions of software system to investigate the relationships between source code metrics and quality parameters ([4] [5] [6]). Consequently, it may hard to comment that their conclusion could be generalized to other version of the systems. In this study, WSDL file of five continuous version of 'ebay' web service are considered as case study for analyzing the effectiveness of the proposed approach. The Java files of all considered WSDL documents are generated using Wsimport tool. Table I displays the details of the experimental datasets. As shown in Table I, the number of classes for all the five versions are more than 1500. The percentage of changed classes from eBay ver. 863 to eBay ver. 865 is 11.61 and the the percentage of change classes from 865 to 867 is 6.77.

[1]$http://developer.ebay.com$

TABLE I: Details of Case study (eBay)

Version	# classes	LOC	# changed classes	# non-changed classes	% changed classes
863	1507	1320264	175	1332	11.61
865	1507	1320576	102	1405	6.77
867	1524	1335460	240	1284	15.75
869	1509	1322904	128	1381	8.55
871	1509	1323080	119	1390	7.89

D. Dependent and Independent Variables

The objectives of this work are to select right set of source code metrics and develop a web-service maintainability prediction model. Maintainability of web-service is defined as "the number of changed lines per Java file' [4]. Hence in the work, Maintainability or change is considered as a dependent variable and set of source code metrics as a independent variables respectively. The dependent and independent variables of the model are shown in Table II.

TABLE II: Effectiveness of metrics

Analysis	**Dependent Variable**	**Independent Variable**
A1	Maintainability	DIT, WMC, NOC, RFC, CBO, NOM, LCOM, MVG, DAC, SIZE1, SIZE2
A2	Maintainability	Reduced feature attributes using feature ranking techniques
A3	Maintainability	Extracted feature attributes using feature subset selection techniques

IV. Analysis of Results

Figure 2 shows our research methodology consisting of various steps for developing web-service maintainability prediction model. Different subset of source code metrics selected using feature selection techniques are used as input of the maintainability prediction model.

The following steps are being followed for finding suitable set of source code metrics to develop a model. In this work, we have considered four types of FR techniques and four types of FSS techniques to select optimal set of source code metrics. These all selected set of source code metrics are applied on five different versions of eBay dataset. Therefore, a total of 135 ((1 considering all features+ 8 feature selection technique) * 5 datasets * 3 different classification technique) distinct classification models are considered in the study.

- **Step 1:** Eleven different source code metrics from the bytecode of the compiled Java files of web service are computed. These metrics are used as a input to develop a maintainability prediction model.
- **Step 2:** Since source code metrics are considered as input to develop a models, the performance of model

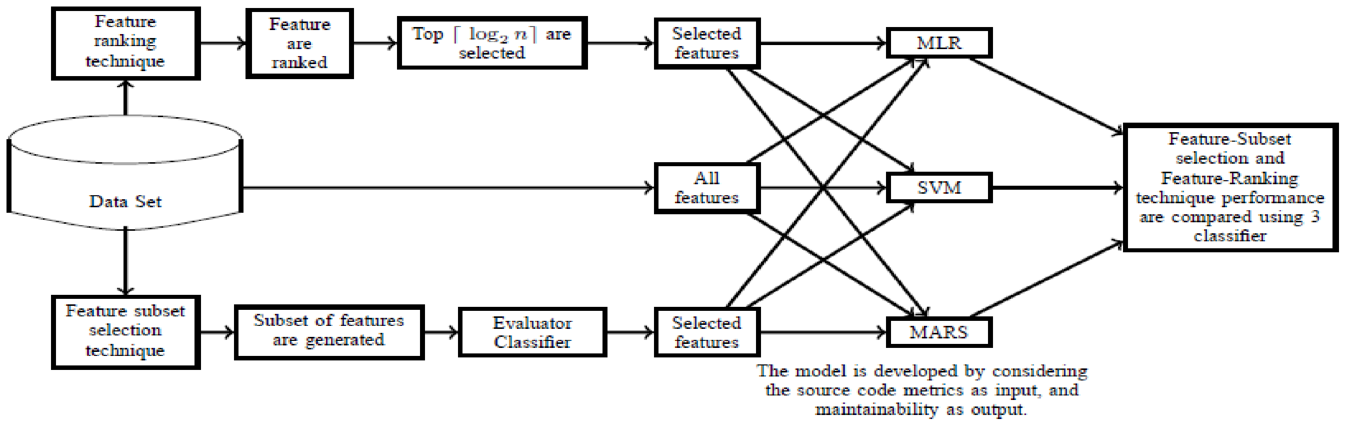
Feature ranking technique
Feature are ranked
Top $\lceil \log_2 n \rceil$ are selected
Selected features
MLR
Data Set
All features
SVM
Feature-Subset selection and Feature-Ranking technique performance are compared using 3 classifier
Feature subset selection technique
Subset of features are generated
Evaluator Classifier
Selected features
MARS
The model is developed by considering the source code metrics as input, and maintainability as output.

Fig. 2: Framework of proposed work

depends on the source code metrics which are used as input to develop a model. In this study, four different FR techniques are applied on all five versions of eBay web service. Each FR technique uses some parameter to sort the features and further top $\lceil \log_2 n \rceil$ ranked features out of n features are used as input to develop a model.

- **Step 3:** Four different FSS techniques have been considered to select optimal set of features.
- **Step 4:** All elven source code metrics, selected set of source code metrics using feature selection techniques are validated using three different classification methods i.e., MLR, SVM, and MARS.

The list of abbreviations as shown in Table III are used in this paper.

TABLE III: Used Naming Conventions for different Techniques

Abbreviation	Corresponding Name
AM	All Metrics
FR1	CS test
FR2	GR Feature Evaluation
FR3	IG Feature Evaluation
FR4	OneR Feature Evaluation
FS1	Classifier Subset Evaluation
FS2	Correlation based Feature Selection
FS3	Filtered Subset Evaluation
FS4	Rough Set Analysis (RSA)

A. Feature ranking technique

In this work, we have considered the use of four different types of FR techniques to select optimal set of features on different version of eBay web service. The selected set of source code metrics after feature ranking methods are tabulated in Table V. Subsequently, these optimal sets of source code metrics as shown in Table V have considered as input to develop web-service maintainability prediction models using MLR, SVM, and MARS techniques. Figures 3 to 5 show the box-plot diagrams for each of the set of source code metrics. The line in the middle of each box represents the median value. The model which has high median value and few number of outliers is the best model for maintainability prediction. Figures 3 to 5, it can be inferred that:

- In case of MLR method, model developed by considering selected set of source code metrics using oneR feature evaluation as input produces better result as compared to others.
- In case of SVM, feature ranking using oneR feature evaluation compute best set of source code metrics for predicting maintainability of web service.
- In case of MARS, model developed by considering selected set of source code metrics using Chi square test as input produces better result as compared to others.

TABLE V: Selected metrics after Feature ranking methods

Method	863	865	867	869	871
FR1	SIZE1, MVG, DAC, SIZE2	DAC, MVG, CBO, LCOM	SIZE1, CBO, DAC, MVG	DAC, LOCM, MVG, SIZE1	DAC, LOCM, WMC, CBO
FR2	DAC, WMC, LCOM, RFC	NOM, WMC, LOCM, DAC	NOM, DIT, MVG, SIZE2	NOM, DIT, RFC, WMC	LOCM, NOM, WMC, RFC
FR3	SIZE1, MVG, SIZE2, CBO	DAC, MVG, LCOM, WMC	SIZE1, RFC, MVG, CBO	DAC, LCOM, MVG, SIZE1	LOCM, WMC, DAC, MVG
FR4	DAC, MVG, NOC, NOM	DAC, MVG, NOC, NOM	MVG, DAC, NOM, LCOM	DAC, MVG, NOC, NOM	DAC, NOC, DIT, CBO

B. Feature subset selection techniques

In this work, we have also considered four types of FSS techniques to select optimal set of features on different versions of eBay web service. The selected set of source code metrics after feature ranking methods are tabulated in Table VI. Subsequently, these optimal sets of source code metrics as shown in Table VI have been considered as input to develop web-service maintainability prediction models. The performance of these developed models are compared using three different parameters. Figures 3 to 5 show the box-plot diagrams for each of source code metrics. Figures 3 to 5, it can be inferred that:

- In case of MLR, model developed by considering selected set of source code metrics using classifier subset evaluation as input produces better result as compared to others.

- In case of SVM, model developed by considering selected set of source code metrics using correlation based feature selection as input produces better result as compared to others.
- In case of MARS, model developed by considering selected set of source code metrics using classifier subset evaluation as input produces better result as compared to others.

TABLE VI: Selected metrics after Feature subset selection methods

Method	863	865	867	869	871
FS1	WMC, DIT, DAC, MVG	WMC, DIT, DAC	DIT, CBO, LCOM, DAC, NOM,MVG	DIT,DAC, NOM	DIT, LCOM,DAC
FS2	DIT, DAC, WMC, SIZE1, NOC	DIT, DAC, WMC, SIZE1, NOC	DIT, DAC, WMC, SIZE1, NOC	DIT, DAC, WMC, SIZE1, NOC	DIT, DAC, WMC, SIZE1, NOC
FS3	DIT, MVG, WMC, DAC	DAC, DIT, WMC	DIT, MVG, LCOM, NOM, CBO, DAC	NOM, DAC, DIT	DAC, DIT, LCOM
FS4	DIT, MVG, DAC, NOM, RFC, SIZE2, SIZE1	DIT, MVG, DAC, NOM, RFC, SIZE2, SIZE1	DIT, MVG, DAC, NOM, RFC, SIZE2, SIZE1	DIT, MVG, DAC, NOM, RFC, SIZE2, SIZE1	DIT, MVG, DAC, NOM, RFC, SIZE2, SIZE1

C. *Multivariate adaptive regression splines (MARS)*

In this experiment, we have considered MARS to develop web-service maintainability prediction model by considering nine various sets of source code metrics as input. Three different performance parameters such as accuracy, precision, and recall have been considered for evaluation of MARS models. In this paper, we also compare the performance of MARS model with performance of the model developed using multivariate linear regression models (MLR), and support vector machine (SVM) methods. The hardware used to carry out our experiments are: Core i5 processor with 4GB RAM and storage capacity of 250GB hard disk. Prediction models are developed using the licensed MATLAB environment at NIT-Rourkela. Table IV, shows the MARS equations for eBay ver. 863. The MARS equations for other versions of eBay web service are of similar type. Figures 3 to 5 show the box-plot diagrams for each of source code metrics. From Figure 3, it can be inferred that, model developed for predicting web-service maintainability using MARS yields better result as compared to MLR, and SVM methods.

D. *Overall performance*

In this study, nine different subset of object-oriented metrics i.e., one set containing all metrics, four sets of source code metrics using FR techniques, and four sets of metrics using FSS techniques are used as a input to develop a model for web-service maintainability model using three different classification technique. Figure 3 to Figure 5 show the box-plot diagrams for each of source code metrics. From the box-plot diagram, it can be inferred that:

- In case of MLR method, model developed by considering selected set of source code metrics using oneR feature evaluation as input produces better result as compared to others.
- In case of SVM, model developed by considering selected set of source code metrics using correlation based feature selection as input produces better result as compared to others.

TABLE IV: MARS Equation for ebay ver. 863

Techniques	Equation
AM	Change =0.59 +0.89*max(0, DAC -0.058) -7.4*max(0, 0.058 -DAC) +54*max(0, 0.058 -DAC)*max(0, NOM -0.033) -4.7e+02*BF4 -0.22*max(0, 1 -DIT) +16*max(0, 1 -DIT)*max(0, 0.033 -NOM) -26*max(0, 0.058 -DAC)*max(0, WMC -0.032) -4.5e+03*max(0, 0.058 -DAC)*max(0, 0.0019 -LCOM) +11*max(0, 0.046 -NOM) -1.2e+02*max(0, NOM -0.046) * max(0, 0.022 -DAC) -2.3*max(0, DAC -0.058)*max(0, CBO -0.27) -9.5*max(0, DAC -0.058)*max(0, 0.27 -CBO) +1e+02* max(0, 0.046 -NOM)*max(0, 0.11 -CBO)
FR1	Change = 0.39 +589.76*max(0, DAC -0.058) * max(0, MVG -0.037) -11192.87*max(0, DAC -0.058) * max(0, 0.037 -MVG) +19*max(0, MVG -0.03) -0.83*max(0, MVG -0.03)*max(0, SIZE1 -0.12) +147.72*max(0, MVG -0.03)*max(0, 0.12 -SIZE1) -590.53*max(0, MVG -0.03)*max(0, DAC -0.028) +444.76*max(0, MVG -0.03)*max(0, 0.028 -DAC) -544.65*max(0, 0.058 -DAC) * max(0, MVG -0.014) +3.8* max(0, DAC -0.058) * max(0, SIZE2 -0.013) -582.59*max(0, DAC -0.058) * max(0, 0.013 -SIZE2) +5.5*max(0, DAC -0.058) * max(0, 0.9 -SIZE1) +1143.24*max(0, 0.03 -MVG) * max(0, DAC -0.0079) -3597.24*max(0, 0.03 -MVG) * max(0, 0.0079 -DAC)
FR2	Change = 0.62 +63* max(0, DAC -0.058) -19* max(0, 0.058 -DAC) -1.2*max(0, LOCM -0.0023) -783.51*max(0, 0.0023 -LOCM) +2.1*max(0, WMC -0.062) +45*max(0, 0.062 -WMC) -18*max(0, WMC -0.062)*max(0, DAC -0.015) -103.44*max(0, WMC -0.062)*max(0, 0.015 -DAC) +122.19*max(0, DAC -0.058)*max(0, RFC -0.96) -75*max(0, DAC -0.058)*max(0, 0.96 -RFC) +35996.08*max(0, 0.0023 -LOCM)*max(0, 0.015 -DAC) +114.11*max(0, 0.062 -WMC)*max(0, DAC -0.024) -11691.98*max(0, 0.062 -WMC)*max(0, 0.024 -DAC) +25*max(0, 0.058 -DAC)*max(0, 0.51 -RFC) -57*max(0, DAC -0.058)*max(0, RFC -0.086) -620952*max(0, LOCM -0.0023)*max(0, 0.051 -WMC)
FR3	Change = 0.32 +29*max(0, SIZE2 -0.07) +4157.52*max(0, SIZE2 -0.07)*max(0, SIZE1 -0.092) -2359.09*max(0, SIZE2 -0.07)*max(0, 0.092 -SIZE1) -38*max(0, SIZE2 -0.07)*max(0, 0.2 -SIZE1) +101.64* max(0, 0.07 -SIZE2) * max(0, CBO -0.24) -0.61*max(0, 0.33 -CBO) +26811.12*max(0, SIZE2 -0.07)*max(0, 0.073 -MVG) -4156.84*max(0, SIZE2 -0.07)*max(0, SIZE1 -0.085) -37* max(0, 0.07 -SIZE2) * max(0, CBO -0.011) -1535.23*max(0, 0.07 -SIZE2) * max(0, 0.011 -CBO) -31264.4*max(0, SIZE2 -0.07)*max(0, 0.074 -SIZE1) +15028*max(0, SIZE2 -0.07)*max(0, 0.067 -SIZE1)
FR4	Change = 0.61 -12*max(0, 0.058 -DAC) +0.15*max(0, NOC +0) +18*max(0, NOC +0)*max(0, 0.11 -CBO) -13*max(0, NOC +0)*max(0, 0.13 -CBO) -2.3*max(0, DAC -0.058) * max(0, CBO -0.13) -18*max(0, DAC -0.058) * max(0, 0.13 -CBO) -1899.53*max(0, 0.058 -DAC)*max(0, 0.012 -CBO) -35*max(0, NOC +0) -0.82*max(0, DIT +0) +1.7*max(0, NOC +0)*max(0, DAC -0.37)
FS1	Change = -2.5 -127.54*max(0, WMC -0.057) +0.25*max(0, DIT +0) +22*max(0, WMC -0.034) +41*max(0, 0.065 -WMC) -6.6*max(0, DIT +0)* max(0, 0.06 -WMC) +138.44*max(0, WMC -0.055) -57*max(0, 0.055 -WMC) -29*max(0, WMC -0.023) +3.1*max(0, 0.89 -WMC)
FS2	Change = 0.52 -11*max(0, 0.058 -DAC) +0.17*max(0, DIT +0) -89*max(0, DIT +0)*max(0, 0.06 -WMC) +47*max(0, 0.023 -CBO) +67*max(0, DIT +0)*max(0, 0.063 -WMC) -305.34*max(0, 0.023 -CBO)*max(0, DAC -0.01) -5284.89*max(0, 0.023 -CBO)*max(0, 0.01 -DAC) -1.1*max(0, DAC -0.058) * max(0, WMC -0.047) +131.29*max(0, DAC -0.058) * max(0, 0.047 -WMC) +19*max(0, DIT +0)*max(0, 0.04 -WMC) -0.73*max(0, NOC +0) -5507.24*max(0, 0.023 -CBO)*max(0, 0.007 -SIZE1)
FS3	Change = 0.59 -2.2*max(0, DAC -0.058) -12*max(0, 0.058 -DAC) -85*max(0, DIT +0) * max(0, WMC -0.06) +20* max(0, DIT +0) * max(0, WMC -0.034) +65*max(0, DIT +0) * max(0, WMC -0.063) -1.2*max(0, DAC -0.058)*max(0, MVG -0.013) -1514.54*max(0, DAC -0.058)*max(0, 0.013 -MVG) +10*max(0, DIT +0) * max(0, 0.039 -MVG) +3.7*max(0, DIT +0) * max(0, DAC -0.37) -0.3* max(0, DIT +0) * max(0, 0.37 -DAC) -1234.42*max(0, 0.058 -DAC)*max(0, 0.024 -MVG) +57*max(0, 0.025 -MVG)
FS4	Change = 0.55 -7.1*max(0, 0.058 -DAC) +33*max(0, 0.058 -DAC)*max(0, NOM -0.033) -531.94*max(0, 0.058 -DAC)*max(0, 0.033 -NOM) -0.21*max(0, 1 -DIT) +14*max(0, 1 -DIT)*max(0, 0.033 -NOM) -6.8*max(0, 0.058 -DAC)*max(0, RFC -0.041) +264.71*max(0, 0.058 -DAC)*max(0, 0.041 -RFC) +12*max(0, 0.046 -NOM) -0.91*max(0, NOM -0.046) * max(0, DAC -0.022) -105.13*max(0, NOM -0.046) * max(0, 0.022 -DAC) -2512.1*max(0, 0.046 -NOM)*max(0, 0.059 -SIZE1) -51*max(0, 0.046 -NOM)*max(0, SIZE2 -0.054) +2631.6*max(0, 0.046 -NOM)*max(0, 0.054 -SIZE2) +1545.4*max(0, NOM -0.046) * max(0, 0.023 -SIZE1)

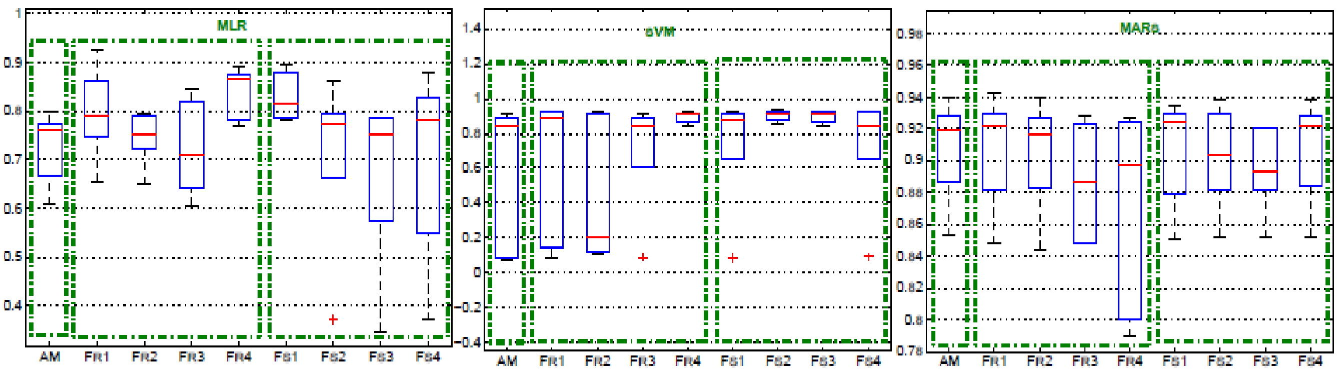

Fig. 3: Accuracy

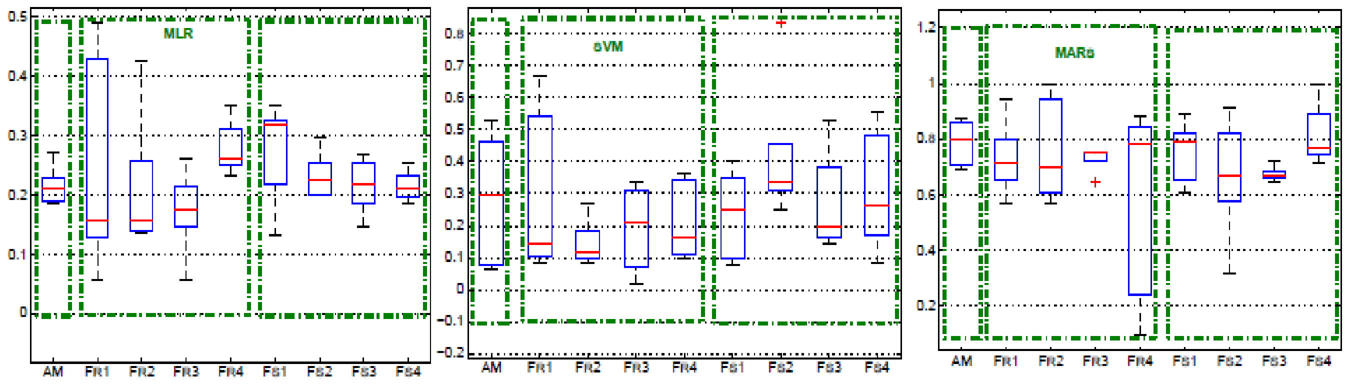

Fig. 4: Precision

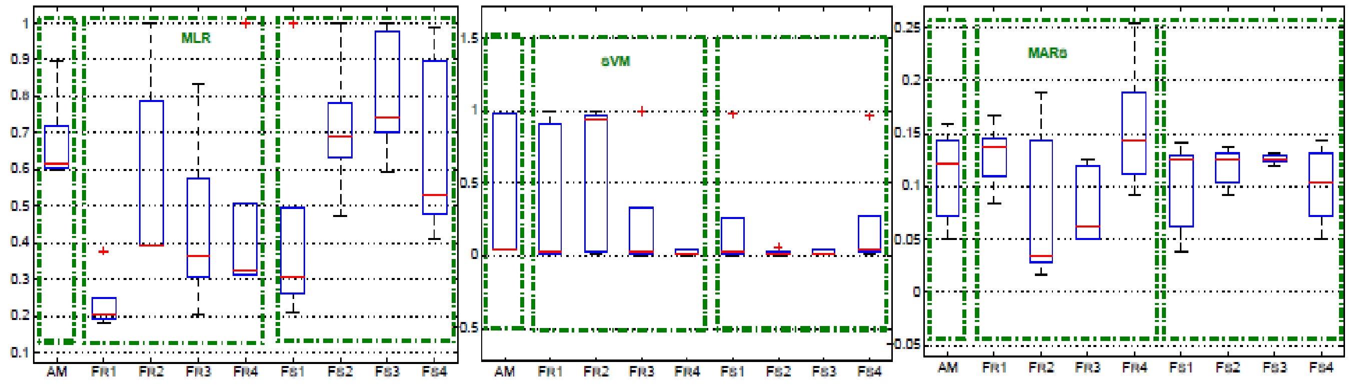

Fig. 5: Recall

- In case of MARS, model developed by considering selected set of source code metrics using classifier subset evaluation as input produces better result as compared to others.
- There exists a small subset of source code software metrics out of total available source code software metrics which are able to predict change-proneness with higher accuracy and reduced value of misclassified errors.

E. Comparison

We use pairwise t-test to compare the performance of feature selection techniques and classification techniques.

- **Feature Selection Techniques:** In this study, nine different set of metrics i.e., one containing all metrics, four sets of source code metrics using FR techniques, and four sets of metrics using FSS techniques have been considered as input to develop a web-service maintainability prediction models. Three

TABLE VII: t-test

(a) t-test: Among different feature selection techniques

Mean Difference									
	AM	FR1	FR2	FR3	FR4	FS1	FS2	FS3	FS4
AM	0	-0.04	0.03	-0.04	-0.14	-0.09	-0.11	-0.09	-0.04
FR1	0.04	0	0.07	0	-0.1	-0.05	-0.07	-0.05	-0.01
FR2	-0.03	-0.07	0	-0.07	-0.16	-0.12	-0.13	-0.12	-0.07
FR3	0.04	0	0.07	0	-0.1	-0.05	-0.07	-0.05	0
FR4	0.14	0.1	0.16	0.1	0	0.04	0.03	0.05	0.09
FS1	0.09	0.05	0.12	0.05	-0.04	0	-0.02	0	0.05
FS2	0.11	0.07	0.13	0.07	-0.03	0.02	0	0.02	0.06
FS3	0.09	0.05	0.12	0.05	-0.05	0	-0.02	0	0.05
FS4	0.04	0.01	0.07	0	-0.09	-0.05	-0.06	-0.05	0
p-value									
	AM	FR1	FR2	FR3	FR4	FS1	FS2	FS3	FS4
AM	-	0.64	0.76	0.5	0.1	0.37	0.2	0.3	0.49
FR1	0.64	-	0.22	0.95	0.19	0.57	0.4	0.53	0.92
FR2	0.76	0.22	-	0.33	0.06	0.26	0.14	0.21	0.34
FR3	0.5	0.95	0.33	-	0.11	0.52	0.28	0.46	0.92
FR4	0.1	0.19	0.06	0.11	-	0.46	0.35	0.16	0.15
FS1	0.37	0.57	0.26	0.52	0.46	-	0.82	0.96	0.58
FS2	0.2	0.4	0.14	0.28	0.35	0.82	-	0.62	0.27
FS3	0.3	0.53	0.21	0.46	0.16	0.96	0.62	-	0.48
FS4	0.49	0.92	0.34	0.92	0.15	0.58	0.27	0.48	-

(b) t-test: Among different Classifier

	Mean Difference			p-value			t-value		
	MLR	SVM	MARS	MLR	SVM	MARS	MLR	SVM	MARS
MLR	0	0.03	-0.15	-	0.58	0	-	0.56	-8.13
SVM	-0.03	0	-0.18	0.58	-	0	-0.56	N-	-3.61
MARS	0.15	0.18	0	0	0	-	8.13	3.61	NaN

(c) t-test: feature ranking Versus feature subset selection techniques (Average)

Mean Difference	p-value	t-value
-0.0362	0.36	-0.9226

different types of classifiers to develop a prediction model considering with three different performance parameters. So for each set of source code metrics, a total number of three set (one for each performance measure) are used, each with 15 data points (3 classifier * 5 dataset). The results of t-test analysis for accuracy performance parameter are summarized in Table VIIa. The results on the other performance parameters are of similar types. One part of the Table VIIa shows the p-value and the other part shows the mean difference values of performance parameter. Table VIIa reveals for most of the cases there is no any significance difference between different approaches as the p-value is greater than 0.05. According to the value of mean difference, oneR feature evaluation yields better result compared to other techniques.

- **Classification Methods:** In this paper, MLR, SVM, and MARS have been considered to develop a maintainability prediction model. This study uses nine different subset of metrics (4 FR techniques + 4 FSS techniques + 1 AM) of five service oriented software dataset with four different performance parameters. So for each prediction technique a total number of four set (one for each performance) are used, each with 45 data point ((8 feature selection method + 1 considering all features) * 5 datasets)). The results of t-test analysis for accuracy performance parameter are summarized in Table VIIb. The results on the other performance parameters are of similar types. Table VIIb reveals that there is a significant difference between different classification techniques as the p-value is lesser than 0.05. According to the value of mean difference, MARS yields better result compared to other techniques.
- **Feature ranking and Feature subset selection methods (Average):** In this study, pairwise t-test has been employed to determine whether feature ranking methods work better than feature subset selection methods or both have performed equally well. The results of t-test analysis for average performance of FR and FSS techniques are summarized in Table VIIc. Since, in this study three different types of classification methods are applied over five service oriented software data sets, so for each feature selection technique a total number of three sets (one for each performance measure) are used, each with 60 data points (4 feature selection techniques * 5 datasets * 3 classifier). Table VIIc reveals that there is no any significant difference between different between FR and FSS techniques as the p-value is greater than 0.05. According to the value of mean difference, FSS techniques yields better result compared to other techniques.

V. Conclusion

We conduct a comparative study of different set source code metrics for predicting web-service maintainability. The effectiveness of the applied set of source code metrics are evaluated using three widely used statistical classification approaches: MLR, SVM, and MARS. The key observations of the study presented in this paper are the following:

- Our experimental analysis reveals existence of a small set of source code metrics from a large number of available source code software metrics across various types which are able to accurately predict maintainability with few mis-classification errors.
- The web-service maintainability prediction model developed using MARS shows better results in comparison to MLR and SVM techniques
- The t-test analysis clearly indicates that for most of the scenarios, there is no any substantial difference between different approaches. This conclusion is a result of the fact that the p-value observed is greater than 0.05. Analyzing the value of mean difference, we can infer that oneR feature evaluation yields better results compared to other techniques.
- From t-test analysis, it is also observed that there is a significant difference between different classification techniques. According to the value of mean difference, MARS yields better result compared to other techniques.
- From experiments, it is observed that the performance of the feature selection methods is varied with the different classification methods used.

Further, work can be replicated on the usage of various neural network models for predicting maintainability. Neural network models over the years have seen an explosion of interest, and applicability across a wide range of problem domain.

Acknowledgment

We are very thankful and grateful to the FIST project of DST (Department of Science and Technology) Government of India for sponsoring our study on web engineering and cloud based computing. The authors are also thankful to the Computer Science and Engineering Department at the NIT Rourkela for providing all the required facilities and guidance to conduct our research

References

[1] Yuming Zhou and Hareton Leung. Empirical analysis of object-oriented design metrics for predicting high and low severity faults. 32(10):771–789, 2006.

[2] Rajendra K. Bandi, Vijay K. Vaishnavi, and Daniel E Turk. Predicting maintenance performance using object-oriented design complexity metrics. *IEEE Transactions on Software Engineering*, 29(1):77–87, 2003.

[3] Ho-Won Jung, Seung-Gweon Kim, and Chang-Shin Chung. Measuring software product quality: A survey of iso/iec 9126. *IEEE software*, 21(5):88–92, 2004.

[4] W. Li and S. Henry. Maintenance metrics for the Object-Oriented paradigm. In *Proceedings of First International Software Metrics Symposium*, pages 52–60, 1993.

[5] Yuming Zhou and Hareton Leung. Predicting object-oriented software maintainability using multivariate adaptive regression splines. *Journal of Materials Processing Technology*, 80(8):1349–1361, 2007.

[6] Ruchika Malhotra and Anuradha Chug. Application of group method of data handling model for software maintainability prediction using object oriented systems. *International Journal of System Assurance Engineering and Management*, 5(2):165–173, 2014.

[7] Lionel C Briand, Jürgen Wüst, John W Daly, and D Victor Porter. Exploring the relationships between design measures and software quality in object-oriented systems. *Journal of systems and software*, 51(3):245–273, 2000.

[8] George Forman. An extensive empirical study of feature selection metrics for text classification. *The Journal of machine learning research*, 3:1289–1305, 2003.

[9] Cesare Furlanello, Maria Serafini, Stefano Merler, and Giuseppe Jurman. Entropy-based gene ranking without selection bias for the predictive classification of microarray data. *BMC bioinformatics*, 4(1):1, 2003.

[10] Jasmina Novakovic. The impact of feature selection on the accuracy of naïve bayes classifier. In *18th Telecommunications forum TELFOR*, pages 1113–1116, 2010.

[11] Robin L Plackett. Karl pearson and the chi-squared test. *International Statistical Review/Revue Internationale de Statistique*, 51(1):59–72, 1983.

[12] Janez Brank, Marko Grobelnik, Natasa Milic-Frayling, and Dunja Mladenic. Consistency-based search in feature selection. *Artificial intelligence*, 151(1):155–176, 2003.

[13] Ron Kohavi and George H John. Wrappers for feature subset selection. *Artificial intelligence*, 97(1):273–324, 1997.

[14] Z. Pawlak. Rough sets. *International Journal of Computer and Information Sciences*, 11(5):341–356, 1982.

[15] S. R. Chidamber and C. F. Kemerer. A metrics suite for Object-Oriented design. *IEEE Transactions on Software Engineering*, 20(6):476–493, 1994.

[16] V. R. Basili, L. C. Briand, and W. L. Melo. A validation of Object-Oriented design metrics as quality indicators. *IEEE Transactions on Software Engineering*, 22(10):751–761, October 1996.

[17] Bingu Shim, Siho Choue, Suntae Kim, and Sooyong Park. A design quality model for service-oriented architecture. In *2008 15th Asia-Pacific Software Engineering Conference*, pages 403–410. IEEE, 2008.

[18] Cristian Mateos, Marco Crasso, Alejandro Zunino, and Jos? Luis Ordiales Coscia. Detecting wsdl bad practices in code–first web services. *International Journal of Web and Grid Services*, 7(4):357–387, 2011.

[19] Mikhail Perepletchikov, Caspar Ryan, and Keith Frampton. Cohesion metrics for predicting maintainability of service-oriented software. In *Seventh International Conference on Quality Software (QSIC 2007)*, pages 328–335. IEEE, 2007.

[20] Mikhail Perepletchikov, Caspar Ryan, Keith Frampton, and Zahir Tari. Coupling metrics for predicting maintainability in service-oriented designs. In *Software Engineering Conference, 2007. ASWEC 2007. 18th Australian*, pages 329–340. IEEE, 2007.

[21] Yijun Yu, Jianguo Lu, Juan Fernandez-Ramil, and Phil Yuan. Comparing web services with other software components. In *Web Services, 2007. ICWS 2007. IEEE International Conference on*, pages 388–397. IEEE, 2007.

[22] José Luis Ordiales Coscia, Marco Crasso, Cristian Mateos, Alejandro Zunino, and Sanjay Misra. Predicting web service maintainability via object-oriented metrics: a statistics-based approach. In *Computational Science and Its Applications–ICCSA 2012*, pages 19–39, 2012.

[23] F. B. E. Abreu and R. Carapuca. Object-Oriented software engineering: Measuring and controlling the development process. In *Proceedings of the 4th International Conference on Software Quality*, volume 186, pages 1–8, 1994.

[24] Thomas J. McCabe. A complexity measure. *IEEE Transactions on Software Engineering*, 2(4):308–320, December 1976.

[25] M.H. Halstead. *Elements of Software Sciencel*. Elsevier Science, New York, USA, 1977.

[26] L. C. Briand, J. W*ü*st, J. W. Daly, and D. V. Porter. Exploring the relationships between design measures and software quality in Object-Oriented systems. *The Journal of Systems and Software*, 51(3):245–273, May 2000.

[27] M. Lorenz and J. Kidd. *Object-Oriented Software Metrics*. Prentice-Hall, NJ, Englewood, 1994.

[28] Shyam R Chidamber and Chris F Kemerer. *Towards a metrics suite for Object-Oriented design*, volume 26. ACM.

[29] José Luis Ordiales Coscia, Marco Crasso, Cristian Mateos, Alejandro Zunino, and Sanjay Misra. Analyzing the evolution of web services using fine-grained changes. In *19th International Conference on Web Services (ICWS), 2012*, pages 392–399, 2012.