Using Source Code Metrics to Predict Change-Prone Web Services: A Case-Study on eBay Services

Lov Kumar
NIT Rourkela, India
lovkumar505@gmail.com

Santanu Kumar Rath
NIT Rourkela, India
skrath@nitrkl.ac.in

Ashish Sureka
ABB Corporate Research, India
ashish.sureka@in.abb.com

***Abstract*—Predicting change-prone object-oriented software using source code metrics is an area that has attracted several researchers attention. However, predicting change-prone web services in terms of changes in the WSDL (Web Service Description Language) Interface using source code metrics implementing the services is a relatively unexplored area. We conduct a case-study on change proneness prediction on an experimental dataset consisting of several versions of eBay web services wherein we compute the churn between different versions of the WSDL interfaces using the WSDLDiff Tool. We compute 21 source code metrics using Chidamber and Kemerer Java Metrics (CKJM) extended tool serving as predictors and apply Least Squares Support Vector Machines (LSSVM) based technique to develop a change proneness estimator. Our experimental results demonstrates that a predictive model developed using all 21 metrics and linear kernel yields the best results.**

***Index Terms*—Change Proneness, Kernel Based Machine Learning, Least Squares Support Vector Machines (LSSVM), Source Code Metrics, Web Services**

I. Introduction

Research Motivation and Aim: Identifying change prone classes or software artefacts are important for a project team so that that they can focus their preventive maintenance efforts such as testing, peer-reviews and source code refactoring on change-prone regions within the software system [1][2]. Web Services are distributed web application components which can be implemented in different languages, deployed on different client and server platforms, are represented by interfaces and communicate using open protocols [3][4]. Web service implementers and providers need to comply with common web service standards so that they can be language and platform independent and can be discovered and used by other applications [3][4]. Web Services are described using a WSDL (Web Service Description Language) document which is an XML format and provides users of the web service with a point of contact [3]. There has been a lot of work in the area of predicting change prone object-oriented software (both close and open source software) using source code metrics [1][2].

However, the problem of change proneness prediction for web service interfaces and WSDL documents is relatively unexplored. Service oriented systems is a different programming paradigm in comparison to object-oriented applications. The development and deployment approach of service oriented systems is different than that of object-oriented systems as the primary focus of service oriented systems is interoperability, platform independence, loosely coupled systems and well-defined interfaces. Romano et al. mention that there are lack of studies that examine indicators of changes for service-oriented system [5].

Coscia et al. mention that there is a high correlation between several traditional (source code-level) OO metrics and the catalog of (WSDL-level) service metrics [6]. The complexity, quality and maintainability, of WSDL interfaces are correlated with WSDL-level service metrics and OO metrics (Point A) [6]. Previous research shows that metrics such as code complexity is correlated to change-proneness of the classes (Point B). Based on Point A and Point B and based on previous work by [6], we infer that there is a correlation between change proneness of service interface and the OO metric implementing the service. Coscia et al. also mention that quality problems in service interfaces can be avoided by applying early code refactoring based on the source code metrics for the program implementing the service [6]. We believe that because service oriented systems are different than object oriented systems, the testing, evaluation and method for predicting change-proneness poses unique challenges and opportunities.

Lov et al. show that the Quality of Service (QoS) characteristics of web services have a correlation with several source code metrics and hence can be estimated by analyzing the source code [7]. Their study demonstrates that the QoS attributes such as maintainability has correlation with source code metrics. Their results shows that the interface complexity of web services varies significantly and is correlated to the source code metrics implementing the services [7]. Several interface complexity attributes such as the data, relation, format, structure, data flow and language are correlated with source code implementing the web-services.

Research Contributions: To the best of our knowledge, the study presented in this paper is the first study on using source code metrics (implementing a web web-service) to predict change-prone web service interfaces (defined using WSDL) using kernel based learning techniques. In context to existing work, our paper makes a novel contribution of presenting the results of an empirical evaluation of the proposed approach

on real world dataset consisting of several version of eBay services. eBay APIs are widely used by developers to build buy and sell based applications using eBay platform. According to the eBay developer program[1], eBay API has more 340000 registered developers. Conducting experiments on eBay web-services which is publicly available and widely used by developers is one of the unique contributions of the study presented in this paper. Furthermore, investigating the performance of Least Squares Support Vector Machines (LS-SVM) which is an extension of the SVM framework for change-proneness prediction of service-oriented systems in novel in context to existing work.

II. Experimental Dataset and Setup

A. Source Code Metrics

We compute 21 source code metrics to develop a predictive model for estimating change-proneness of web services. These source code metrics are computed using CKJM extended tool. CKJM extended[2] is an extended version of the CKJM tool for calculating Chidamber and Kemerer Java Metrics and several other metrics such as WMC, LCOM, CBO, MFA, CAM and CC [8]. We use Chidamber and Kemerer Java Metrics in our study as these metrics have been widely used for predicting change-proneness of object oriented system and has tool support for the purpose of collecting the relevant metric data from source code.

B. Data Collection and Preparation

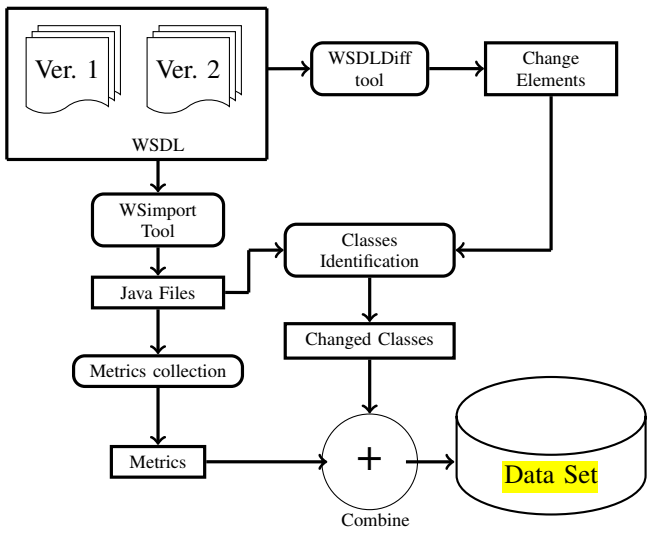


Fig. 1. Data Collection and Preparation Process

Figure 1 displays the multi-step process of data collection and preparation. We download the eBay web services from a publicly available source[3] so that our experiments can be replicated and used for benchmarking and comparison. The

TABLE I
Experimental Dataset Details of eBay Web Services

Version	# Classes	LOC	# Changed	# Non-Changed	% Changed
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

eBay Trading API is an active project and consists of several versions. The latest version as of 17 June 2016 is 973. We use 5 versions of the API in our experiments (refer to Table I). We use WSDL Diff tool for comparing subsequent versions of WSDL interfaces [9]. The WSDL Diff tool automatically extracts the changes in subsequent versions of WSDL interfaces. Then we use the wsimport[4] tool to parse WSDL document file of a Web Service and generate its corresponding Java class (extracting the Java source code implementing the service). As shown in Figure 1, the next steps consist of computing the source code metrics using CKJM extended and LOC metrics tools. The Java classes which contains changed (addition, deletion and modification) elements of the Web Service such as (Operation, Message, XSDType) are termed as changed classes. Table I displays the experimental datasets details. As shown in Table I, the number of classes for all the 5 versions are more than 1500. The percentage of change classes from version 865 to 867 is 15.75 and the percentage of change classes from 869 to 871 is 7.89. Table I reveals that we conduct our experiments on a system with more than one million lines of code. The experiments are conducted on a long-lived, large and complex software system. The change made to a Class can be small (for example, just one line of code) or large (modifying more than 50 or 100 lines of code). However, for the study presented in this paper, we consider all the changes as equivalent and do not differentiate between type of changes (a bug fix for a feature enhancement or refactoring) or the size of the change. The dataset was imbalanced as the classes are not represented equally. The ratio of Changed and Non-Changed classes is 1:11 for the version 869. We address the imbalance training problem by conducting experiments on 5 versions and using the F-measure metrics.

III. Research Questions

Based on our analysis of previous work, we observe that mostly researchers have used only two different versions of a software to compute the change-prone classes investigate the relationships between change-proneness and source code metrics [10][11]. We believe conclusions based on such studies cannot be generalized for cases when we have multiple versions of the systems. In this paper, WSDL file (metric values) of five continuous version of eBay Web service are considered for experiments making the results more generalizable.

[1]https://go.developer.ebay.com/2016-fall-update
[2]http://gromit.iiar.pwr.wroc.pl/p_inf/ckjm/
[3]http://developer.ebay.com/Devzone/XML/docs/ReleaseNotes.html
[4]http://docs.oracle.com/javase/6/docs/technotes/tools/share/wsimport.html

RQ1: Is it possible to predict change-proneness of web-services (defined using a WSDL document) using source code metrics implementing the web services? What is the performance of 21 CKJM metrics in-terms of their predictive power?

RQ2: What is the variation in performance (measured in-terms of accuracy and F-measure) of several LS-SVM classification models over different set of source code metrics? Do we see different performance for different (linear, polynomial and RBF) LS-SVM kernel functions?

IV. Empirical Validation of Software Metrics

Following are the various steps involved in applying Least Squares Support Vector Machine (LS-SVM) based technique for evaluating our proposed approach consisting of computing source code metrics implementing a web-service and using it as predictor variables to estimate change proneness.

Data Set Collection and Ground-Truth Creation: The first step in the solution approach consists of creating a ground-truth. Source code metrics are computed for all the Java files and for all the 5 versions. The WSDL document files which were changed between subsequent versions are identifies using the WSDL Diff tool. A ground-truth (labeled dataset) is thus created. The ground truth is used for the supervised learning task using LS-SVM technique.

Data Normalization We apply feature scaling and data normalization to bring all source code metrics between the range of 0 to 1 i.e., [0 1]. We apply the min-max normalization technique which performs a linear transformation on the original source code metric values such that the relationships among the original data values are preserved. Hence, before applying the machine learning algorithm and subsequent t-test analysis, the input metrics are normalized or standardized using min-max scaling.

t-test Analysis: We conduct a t-test analysis to understand whether the means of two groups (change proneness metric group and not-change proneness metric group) are statistically different from each other or not. Metrics with significant difference between their means are considered and the metrics having no significant difference between their means are rejected. We check for acceptance and rejection of null ($H0$) hypothesis. $t-test$ on each metric has been applied and compared with their corresponding $p-value$ for each metric as a measure of how effectively the metrics separate the groups. Metrics having $p-value$ lesser than 0.05 have strong discrimination power.

Univariate Logistic Regression (ULR) Analysis: Binary univariate logistic regression (ULR) analysis is applied to test the significance of correlation between source code metrics and change-proneness. We perform ULR on each selected metrics to investigate whether the source code metrics were significant predictors of change-proneness or non change-proneness classes or not. We consider only those metrics having their coefficient p-value less than 0.05. Based on t-test and ULR analysis, we accept and reject hypothesis mention in Table IV.

Correlation Analysis between Metrics: We compute Pearson product-moment correlation coefficient between 21 metrics. Our objective is to compute the linear correlation between the metrics. When the r-value (correlation coefficient) between two source code metrics is greater than 0.7 i.e., $r \geq 0.7$ then there exists a strong positive correlation between both the metrics. Similarly, When the r-value is less than -0.7 i.e., $r \leq -0.7$ then there exists a strong negative correlation between both the metrics.

Multivariate Linear Regression Stepwise Forward Selection: Eliminating the insignificant metrics and collinearity among the metrics does not necessarily mean that we have the right set of source code metrics for change-proneness prediction. To select right set of source code metrics, we consider multivariate linear regression stepwise forward selection process.

Validation of Selected Source Code Metrics: After finding the right set of source code metrics, change-proneness models are developed to validate the capability of selected set of source code metrics for change-proneness prediction. The construction of investigation models is carried out on five different dataset. In this work, LSSVM with three different types of kernel functions are considered for development of prediction models. The performance of all prediction models are compared using two different performance evaluation parameters: Accuracy and F-measure.

TABLE II
Hypothesis (H) - Change Proneness Prediction Metrics (M)

H	M	H	M	H	M	H	M
H1	WMC	H2	DIT	H3	NOC	H4	CBO
H5	RFC	H6	LCOM	H7	Ca	H8	Ce
H9	NPM	H10	LCOM3	H11	DAM	H12	MOA
H13	MFA	H14	CAM	H15	IC	H16	CBM
H17	AMC	H18	LOC	H19	SLOC-P	H20	SLOC-L
H21	MVG	-	-	-	-	-	-

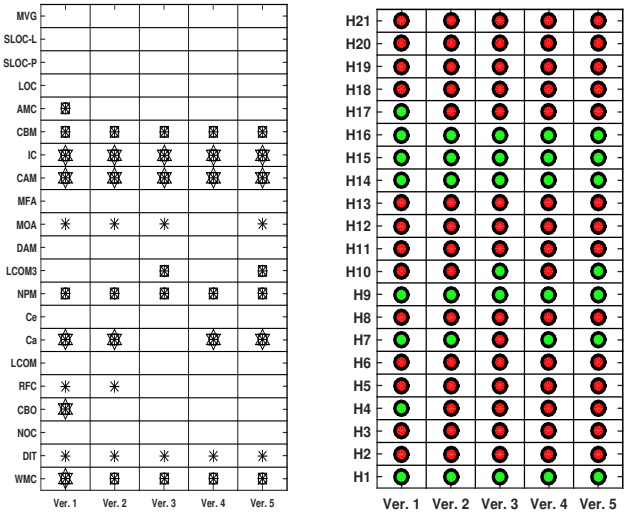
V. Experimental Results

A. Source Code Metrics Validation

We followed four different steps to validate the source code metrics. In each steps, we eventually select some of source code metrics (called as SM or Selected Metrics) among available source code metrics based on the output of feature selection method. Figure 2a shows the selected set of source code metrics in each steps for all five version of eBay web service. The graphs are represented using four different symbols as described below. The Selected Metrics (SM) are different for each version and displayed using the hexagonal

with square, circle and star in Figure 2a. All Metrics (AM) comprises of all the 21 metrics shown in Figure 2a.

1) Star ($*$): source code metrics selected after t-test analysis.
2) Circle with star (✇): source code metrics selected after t-test and ULR analysis.
3) Square with circle and star (✈): source code metrics selected after t-test, ULR analysis and cross correlation analysis.
4) Hexagonal with square, circle and star (❁) : source code metrics selected after t-test, ULR analysis, cross correlation analysis and MLR stepwise forward selection method.



(a) Selected Set of Metrics (b) Hypothesis

Fig. 2. Source Code Metrics Validation

t-test Analysis: The first step of source code metrics validation is to test the statistical significance between change and non-change-proneness group source code metric. Here, we apply $t-test$ on each source code metric and consider $p-value$ of each source code metrics to measure *how effectively it separates the change or non-change-proneness groups*. The metrics having $p-value$ lesser than 0.05 have strong discriminatory power. The result of t-test analysis is shown in Figure 2a.

The $p-value$ are represented using two different symbols (blank box (□): $p-value > 0.05$ and box with star ($*$): $p-value <= 0.05$). The metrics having $p-value$ lesser than 0.05 are significant differentiators of the change or non-change-proneness classes. From Figure 2a, we can infer that WMC, DIT, CBO, RFC, Ca, NPM, MOA, CAM, IC, CBM, and AMC source code metrics significantly differentiate the change or non-change-proneness classes for eBay version 863. Therefore, we accept the hypothesis H1, H2, H4, H5, H7, H9, H12, H14, H15, H16, H17 and conclude that these metrics are significantly differentiators of the change or non-change-proneness classes for eBay version 863.

Univariate Logistic Regression (ULR) Analysis: Univariate Logistic Regression (ULR) analysis is used to investigate whether the selected set of source code metrics using $t-test$ analysis are significant predictors of change-pronene classes or not. A source code metrics is significant predictor of class change-proneness if its p-value of coefficient is less than 0.05. The source code metrics having $p-value$ values of coefficient lesser than 0.05 are shown using circle with star (✇) in Figure 2a.

From Figure 2a, we interpret that among WMC, DIT, CBO, RFC, Ca, NPM, MOA, CAM, IC, CBM, AMC only WMC, CBO, Ca, NPM, CAM, IC, CBM, AMC source code metrics are significant predictors of change-proneness for eBay web service version 863. Hence, we accept the H1, H4, H7, H9, H14, H15, H16, H17 hypothesis and conclude that these source code metrics are significantly associated with class change-proneness. Based on t-test and ULR analysis, we accept and reject hypothesis mentioned in Table IV. Figure 2b shows the acceptance and rejection of hypotheses for all five version of eBay web service. The vertical and horizontal axis shows the name of hypothesis and eBay version respectively. The acceptance and rejection of hypothesis depicted using green circle (●) and red circle (●) respectively.

Correlation Analysis between Metrics: Figure 3 shows the Pearson's correlation among the all source code metrics for eBay service version 863. The result for other versions are similar. We use four different symbols as described below:

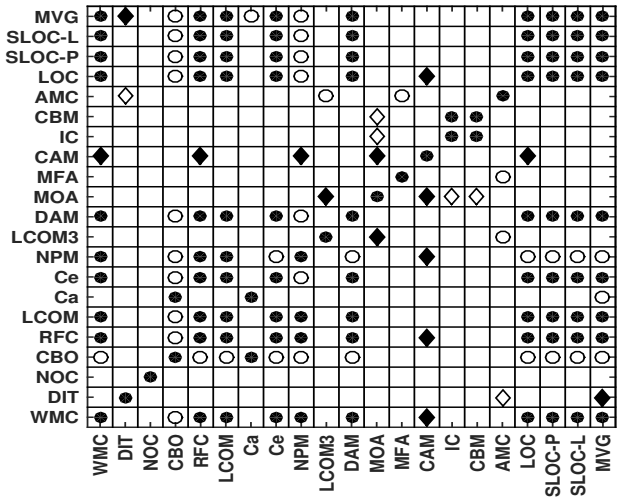


Fig. 3. Correlation between source code metrics

1) Black circle (●): r value between 0.7 and 1.0 indicate a strong positive linear relationship.
2) white circle (○): r value between 0.3 and 0.7 indicate a weak positive linear relationship.
3) Black Diamond (◆): r value between -1 and -0.7 indicate a strong negative linear relationship.
4) white Diamond (◇): r value between -0.7 and -0.3 indicate a weak negative linear relationship.

5) Blank circle: no linear relationship.

In this work, cross correlation analysis are performed only on significant set of source code metrics identified using t-test and ULR analysis. If a significant source code metric shows higher correlation i.e., r-value >=0.7 or r-value <=-0.7 with other significant source code metrics, then we check the performance of these source code metric individually and in combination for change-prone prediction and select a metric or group of metrics, whomsoever perform better. The selected set of source code metrics after cross correlation analysis are shown using ⊠ in Figure 2a.

Multivariate Linear Regression Stepwise Forward Selection: The selected set of source code metrics obtained after cross correlation analysis does necessarily mean that we have a suitable set of source code metrics for change-proneness model development. In this study, multivariate linear regression stepwise forward selection method is considered to select right set of source code metrics for development of change-proneness models. The best set of source code metrics after all four analysis i.e., t-test, ULR analysis, Cross Correlation analysis and multivariate linear regression stepwise forward selection method are shown using ⊠ in Figure 2a.

B. Least Square Support Vector Machine (LSSVM) Classifier

In this work, consider least square support vector machine (LSSVM) classifier with three different kernel methods for validating the selected set of source code metrics. We apply 20 fold cross-validation for comparing the predictive models. 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 licenced *MATLAB* environment at NIT-Rourkela [5]. The performance of each prediction model is evaluated in terms of two performance parameters i.e., accuracy and F-Measure. The x-axis in Figures 4 to 6 represents the five versions of the eBay web-service.

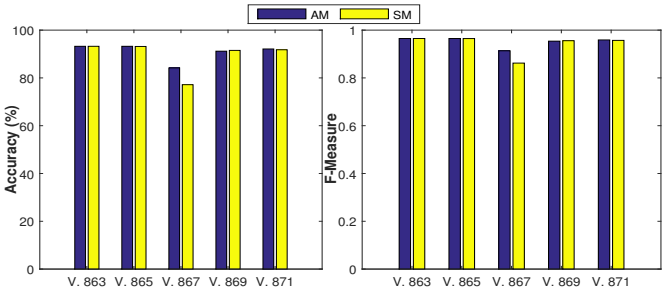


Fig. 4. Results of the validation of prediction models constructed using selected subset of metrics and all metrics (Linear Kernel)

The y-axis represents the accuracy and f-measure performance evaluation metrics. The two bars represents the performance for the AM and SM metrics respectively. Figures 4 to 6 show the obtained performance values for different version of

[5]http://www.nitrkl.ac.in

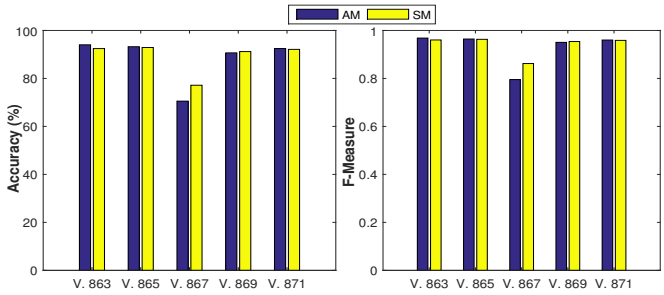


Fig. 5. Results of the validation of prediction models constructed using selected subset of metrics and all metrics (Polynomial Kernel)

eBay web service using LSSVM with linear, polynomial and radial basic function. From Figures 4 to 6, it can be observed that change-proneness model developed using selected set of source code metrics passes the desired prediction accuracy and comparable with the models that are build by considering all twenty one metrics. This results confirm that the ability of these selected source code metrics to predict change-proneness in the eBay web service.

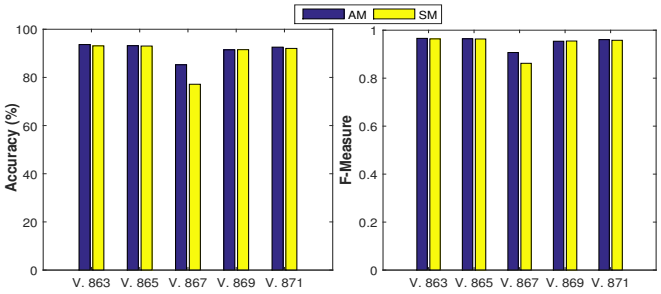


Fig. 6. Results of the validation of prediction models constructed using selected subset of metrics and all metrics (RBF Kernel)

C. Comparison of Results

We apply pairwise t-test to analyze the performance of source code metrics validation techniques and different kernels. We use the R statistical computing software to analyze our data. We conduct a multiplicity of t-tests on more than two samples and does a comparison of more than two means. The results of t-test analysis for performance parameters are summarized in Table V-C.

Source Code Metrics Validation: LSSVM with three different types of kernel methods have been considered to develop a prediction model considering two different performance parameters i.e., Accuracy, and F-Measure. For each set of of source code metrics (all metrics and selected set of metrics) two sets (one for each performance measure) are used, each with 15 data points (3 classifier * 5 dataset). The results of t-test analysis for performance parameters are summarized in Table IVa.

Table IVa contain three parts. The first part of the table IVa shows the p-value of performance parameters, second

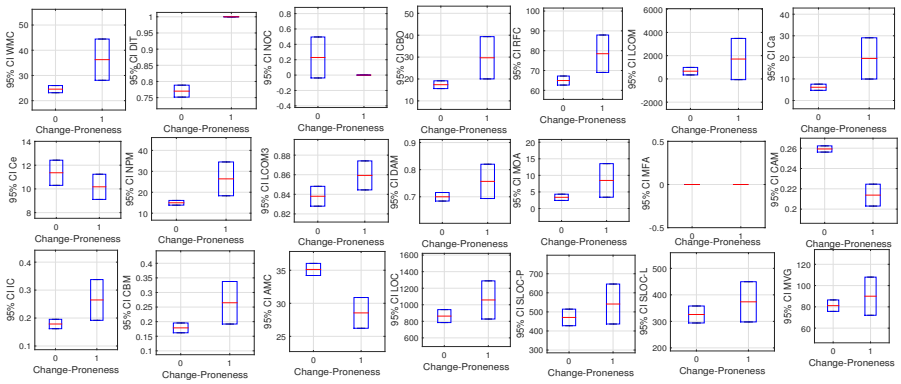


Fig. 7. Error Box Plot Diagrams for eBay Version 863 for all 21 Metrics showing Statistically Significant Difference between the Change-Proneness and Non-change Proneness Groups

part shows the t-value and last part contains the mean difference values between different pairs. From Table IVa it is evident that there is no significant difference between these approaches due to the fact that p-value greater than 0.05. However, by carefully inspecting the value of mean difference, we observe that the model developed using all metrics resulted in sightly better (only 0.57 % higher accuracy) results compared to models using selected set of metrics.

Metrics Correlation with Change-Proneness: We create an error box plot diagrams to gain more insight into the relationship between change-proneness and non-change proneness classes. These box plot diagram (ref Figure 7) helps us to cross-check the results of our t-test analysis. If the means and their confidence intervals do not overlap, then we will find a statistically significant difference between the change-proneness and non-change proneness groups otherwise these groups are probably not significantly different from one another. Figure 7 shows an error box plot diagram of the means and the 95% confidence intervals for all the metrics of eBay 863. We observe similar results for other versions of the eBay. From Figure 7, we can observe that the boxes of WMC, DIT, CBO, RFC, Ca, NPM, MOA, CAM, IC, CBM, AND AMC metrics do not overlap (i.e. are significantly different). The mean (red line) in the middle of the box for the change-proneness groups are higher than non-change-proneness groups. Based on these box plot diagrams, we accept hypotheses H1, H2, H4, H5, H7, H9, H12, H14, H15, H16, and H17, concluding that these metrics are capable of identifying the change-proneness and non-change proneness classes for ebay 863 which subsequently impacts the interfaces.

Kernel Methods: In our study, LSSVM method with three different kernel methods are considered to develop a model to predict class change-proneness. Suykens et al. mention that LS-SVM is comparable to SVM in terms of generalization performance based on extensive empirical studies conducted by several researchers [12]. Several types of kernels such as linear, polynomial and Radial Basis Function (RBF) can be used with both SVM and LS-SVM. Not only generalization performance, but LS-SVM has less computational complexity in comparison to SVM as LS-SVM can be efficiently estimated using iterative methods. The computational efficiency in LS-SVM is due to the linear system in comparison to the quadratic problem in SVM [12].

Generalization performance and computational complexity are important design decisions in our proposed approach and hence we select LS-SVM based machine learning technique. One of the main advantages of SVM-based methods over several machine learning techniques is that LS-SVM leads to global models that are often unique and nonlinear regression can be performed easily as an extension to linear regression[6]. Our study uses two different subset of metrics (metrics selected using proposed framework + 1 considering all features) of five different version of eBay web service with two different performance parameters are considered. So for each kernel method a total number of two sets (one for each performance) are used, each with 10 data point ((1 selected set of source code metrics + 1 considering all features) * 5 datasets)). The results of t-test analysis for performance parameters is summarized in Table IVb.

From Table IVb, it is evident that, there is no significant difference between these approaches, due to the fact that the p-value is greater than 0.05. However, upon closely inspecting the value of mean difference, LSSVM with linear kernel function yields better results compared to other kernels.

VI. Answers to Research Questions

RQ1 Figures 4 to 6 and Table V-C reveals the performance of selected set of source code metrics and all metrics. From Figures 4 to 6 and Table V-C, we infer that it is possible to predict change-proneness of web services (defined using a WSDL document) using source code metrics implementing the web services. Figure 2 shows the predictive power for the various source code metrics. Figures 4 to 6 reveals that

[6]http://pubs.acs.org/doi/abs/10.1021/ac035522m

TABLE III
RESULTS OF T-TEST STATISTICAL SIGNIFICANCE TESTING

Accuracy						
	P-value		**t-value**		**Mean Difference**	
	AM	SM	AM	SM	AM	SM
AM	1.00	0.80	0.00	0.25	0.00	0.56
SM	0.80	1.00	-0.25	0.00	-0.56	0.00
F-Measure						
	P-value		**t-value**		**Mean Difference**	
AM	1.00	0.99	0.00	0.01	0.00	0.00
SM	0.99	1.00	-0.01	0.00	0.00	0.00

(a) Feature Selection Techniques

Accuracy									
	P-value			**t-value**			**Mean Difference**		
	Lin	Poly	RBF	Lin	Poly	RBF	Lin	Poly	RBF
Lin	1.00	0.56	0.97	0.00	0.59	0.04	0.00	1.71	0.09
Poly	0.56	1.00	0.60	-0.59	0.00	-0.54	-1.71	0.00	-1.62
RBF	0.97	0.60	1.00	-0.04	0.54	0.00	-0.09	1.62	0.00
F-Measure									
	P-value			**t-value**			**Mean Difference**		
Lin	1.00	0.41	0.72	0.00	0.84	0.36	0.00	0.02	0.00
Poly	0.41	1.00	0.59	-0.84	0.00	-0.55	-0.02	0.00	-0.01
RBF	0.72	0.59	1.00	-0.36	0.55	0.00	0.00	0.01	0.00

(b) Classification Methods

the accuracy of the predictive models are above 80% for all the three types of Kernel and for all the five version of the metrics. Furthermore, a consistent accuracy of above 80% for both the metrics set (AM and SM) is an evidence of the effectiveness of the proposed approach.

RQ2: From Figures 4 to 6, conclude that the performance of the LSSVM method varies with the different set of source code metrics. This result shows that selection of classification metrics to develop a model for predicting change-proneness classes is affected by the selection of source code metrics. Similarly, we observe a variation in performance based on the kernel functions used. Figures 4 to 6 reveals that the AM metrics outperform the SM metric for the linear kernel and RBF kernel. However, the AM metric does not dominate SM metric for the polynomial kernel. We believe it is important to understand the extent of predictive accuracy difference between AM and SM to do a cost-benefit analysis and hence we presented a detailed results on both the set of metrics.

Threats to Validity: The number of versions of eBay web services are also limited. While we conduct experiments on one real-world data, results on limited dataset is one of the threats to validity.

VII. CONCLUSION

We conclude that it is possible to predict change proneness of WSDL documents and services using source code metrics and kernel based learning techniques. We conclude that the model developed using all metrics results in sightly better (only 0.57% higher accuracy) results compared to models on selected set of metrics. The predictive model developed using LS-SVM linear kernel yields better results compared to other kernels. The performance of the selected set of source code metrics varies with the different classification methods (such as linear, polynomial and RBF kernel). Even after removing 85.71% (Average) of the available number of source code metrics, the developed change-proneness prediction models were not adversely affected; in fact, in some of the cases the results were better.

REFERENCES

[1] A Güneş Koru and Hongfang Liu. Identifying and characterizing change-prone classes in two large-scale open-source products. *JSS*, 80(1):63–73, 2007.

[2] Ruchika Malhotra and Megha Khanna. Investigation of relationship between object-oriented metrics and change proneness. *IJMLC*, 4(4):273–286, 2013.

[3] Francisco Curbera, Matthew Duftler, Rania Khalaf, William Nagy, Nirmal Mukhi, and Sanjiva Weerawarana. Unraveling the web services web: an introduction to soap, wsdl, and uddi. *IEEE Internet computing*, 6(2):86, 2002.

[4] Eric Newcomer and Greg Lomow. *Understanding SOA with Web services*. Addison-Wesley, 2005.

[5] Daniele Romano. Analyzing the change-proneness of service-oriented systems from an industrial perspective. In *ICSE*, pages 1365–1368, 2013.

[6] 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.

[7] Lov Kumar, Santanu Rath, and Ashish Sureka. Predicting quality of service (qos) parameters using extreme learning machines with various kernel methods. In *Workshop on Quantitative Approaches to Software Quality (QuASoQ 2016) co-located to (APSEC 2016)*. CEUR, 2016.

[8] Shyam R Chidamber and Chris F Kemerer. *Towards a metrics suite for Object-Oriented design*, volume 26. ACM, 1991.

[9] José Luis Ordiales Coscia, Marco Crasso, Cristian Mateos, Alejandro Zunino, and Sanjay Misra. Analyzing the evolution of web services using fine-grained changes. In *ICWS*, pages 392–399, 2012.

[10] Yuming Zhou and Hareton Leung. Predicting object-oriented software maintainability using multivariate adaptive regression splines. *JMPT*, 80(8):1349–1361, 2007.

[11] 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.

[12] Johan AK Suykens, Lukas Lukas, and Joos Vandewalle. Sparse least squares support vector machine classifiers. In *ESANN*, pages 37–42. Citeseer, 2000.