Inter Project Validation for Change Proneness Prediction using Object Oriented Metrics

Dr. Ruchika Malhotra#, Megha Khanna*

#Department of Software Engineering, Delhi Technological University
Bawana Road, Delhi- 110042, India
ruchikamalhotra2004@yahoo.com
**AcharyaNarendraDev College*
University of Delhi, Govindpuri, Kalkaji, New Delhi- 110019, India
meghakhanna86@gmail.com

Abstract- **Developing prediction models for determining change prone classes of a software is a significant upcoming research area. Such models help usin effective and efficient resource utilization during maintenance phase. A crucial step in developing these prediction models is the use of training data of that project. Thus training data of a project dictates its prediction model. This dependency leads to difficulties in applying the prediction model to other projects. The training data of a project, consists of metrics data as well as change statistics. Although computation of metrics data is easy, but collection of change statistics is complex. In this paper we attempt to reuse the generated prediction model of one project and validate it on another project. For the purpose of our evaluation, we have used two open source projects written in Java language. The performance of the predicted models was evaluated using Receiver Operating Characteristic (ROC) analysis. The results of our study indicate that we can successfully apply the training sets for inter project validation. These results help us in optimizing time and effort required to generate training set of each project. It leads to efficient utilization of constraint resources and time.**

Keywords- **Change Proneness, Empirical validation,Inter Project validation,Object Oriented Metrics, Open Source, Receiver Operating Characteristic (ROC) analysis.**

1. INTRODUCTION

The principal challenge faced by software industry today, is efficient utilization of limited resources like time, cost and effort. Over the years, various researchers have conducted extensive studies to interrogate the relationship between object oriented (OO) metrics and software quality attributes like fault proneness and maintainability [1-5]. These studies assist us in effective and competent utilization of limited resources.

In order to establish and uncover the relationship between software quality attributes and OO metrics, we need prediction methods. An important step in developing prediction models is the learning process. With the help of learning, a prediction model adapts and adjusts its performance based on empirical data. The empirical data needed for learning process is called the training set. It helps in ascertaining possible predictive relationships between the dependent and independent variables. A training set consists of an input vector and an answer vector, which is used with a supervised learning method to train a knowledge database [6]. The training set is often used in alliance with a test set,which is used to examine the stability and fitness of the predictive relationship [6].

The training set for prediction model of a particular software, incorporates in it various characteristics of a software like its domain, environment, language etc [7]. Thus, a prediction model is highly dependent on its training set. However, possible reuse of training set of a particular project on another project could be of great advantage, as computation of training set involves a lot of effort. Such reuse of training set of a project to validate another project is called inter project validation. Our study aims at reusing training data for change proneness prediction for inter project validation.

Change proneness, i.e. the probability that a particular part of the software would undergo change in future, is a significant software quality attribute [8-10]. If a class is predicted as change prone, we can track its changes and test it rigorously leading to better quality software. Prediction of change prone classesusing OO metrics leads to effective resource utilization during maintenance [10].

The paper investigates the following research goals: (1) How accurately and precisely can predicted results of one project be applied on another project i.e. how successful is inter project validation? (2) What is the relationship between OO metrics and change proneness? (3) Which metrics are effective indicators of change proneness? (4) What is the accuracy of machine learning methods?
For the purpose of our analysis, we have taken two open

source software (Frinika and FreeMind) written in Java language. Two versions of both the software were taken and analyzed for changes. The changes were counted in terms of added, deleted and modified lines in the recent version classes with respect to previous version classes. OO metrics were also generated for all the classes of both the software. OO metrics and change statistics combined to yield data points. These data points form the training data of a project.

In order to evaluate inter project validation, we used Frinika software as our training set and validated it on Freemind software (test set). Apart from this, we also evaluated the results of Freemind using ten-fold cross validation.Both results were analyzed and yielded comparable results. The results were analyzed using Receiver Operating Characteristic (ROC) analysis. Though inter project validation for fault proneness has been investigated earlier [7], but to the best of our knowledge, no previous research has been conducted that validates inter-project prediction using change proneness data.

The main findings of our work are(1) Metrics CBO, RFC and LDC (refer Table 1) are important indicators for prediction of change prone classes (2) inter-project reuse of training data is possible for prediction of change prone classes using OO metrics and yields comparable results(3) the predicted results would help in efficient planning of testing resources during maintenance.
This paper is organized as follows, Section 2 summarizes related work. Section 3 explains the independent and the dependent variable. Section 4 specifies empirical data collection method. Section 5 enumerates the research methodology. The results of the study are presented in Section 6. Section 7 states threats to validity while Section 8 presents conclusion of the work.

2. RELATED WORK

Change proneness is a critical software quality attribute [8-10]. Han et al. [11]contributed in improvement of quality of design. TheyevaluatedJfreeChart, an open source project by applying Behavioral Dependency measurement (BDM). BDM is an approach to rate classes according to their probability of change.They concluded that BDM is a significant criteria for prediction of change proneness.
According to Ambros et al. [12] change coupling is defined as the dependency of various modules of a software on one another as they change and transform together. They analyzed the relationship between change coupling and software defects using correlation and regression analysis. They validated their study on three large software systems and found change coupling to be correlated more with large defects as compared to minor ones.

Sharafat et al. [9] analyzed the change history and source code of a class to predict the probability of change in the class. Reverse engineering techniques were used to analyze source code over various software releases and to collect code metrics. They also investigated the probability that a change would propagate a change in another class by studying the dependencies obtained from UML diagrams of the system.JFlex, a lexical analyzer generator for Java was used by them for evaluating their model.

Zhou et al. [8] studied Eclipse, a software developed in java programming language by analyzing three size metrics SLOC (Source lines of Code), NMIMP (Number of Methods Implemented in a class) and NumPara (sum of the number of parameters of the methods implemented in a class) to find out the confounding effect of class size on the relationship between OO metrics and change proneness.A confounding variable is the one which leads to distortion of the actual relationship between the dependent and the independent variable. They concluded that confounding effect of class size exists on the relationship between OO metrics and change proneness.

Watanabe et al. [7] have tried to evaluate the reuse of a fault prediction model on a C++ and Java open sourceproject. They also suggested compensation techniques to support inter language reuse of prediction models. They demonstrated inter language as well as inter project reuse of training data to cut costs and increase accuracy of results. Zimmermann et al. [13] studied cross-project defect prediction on a huge scale by analyzing 12 real-world applications for 622 cross-project predictions. They concluded that cross project prediction using models from the projects in the same domain or with the same process are not always successful. They investigated and determined factors that influence cross –project predictions. They also derived decision trees to scrutinize the precision, recall and accuracy before attempting a prediction.

Malhotra et al. [10] investigated the relationship between OO metrics and change proneness of a class. For the purpose of their analysis they chose three open source software written in java language and used statistical as well as machine learning methods for predicting change proneness using ROC (Receiver Operating Characteristics) model. They concluded that relationship between OO metrics and change proneness of a class exists and machine learning methods outperformed statistical methods in this evaluation.

Lu et al. [14] examined 102 java systems by using statistical meta-analysis technique to ascertain the ability of 62 OO metrics to predict change proneness. They concluded that size metrics exhibit moderate, coupling and cohesion exhibit lower than size metrics and inheritance exhibits poor capability to predict whether a class is change prone or not. They also ascertained using sensitivity analysis that their results are not biased on data selection.

3. INDEPENDENT AND DEPENDENT VARIABLE

In this section, we explain the independent and dependent variable in this study.

3.1 Independent Variable

The independent variable i.e. various OO metrics collected for each class of the software are summarized in Table 1. Several metrics are used in our study to account for various characteristics of software like size, coupling, cohesion, inheritance etc. The value for these OO metrics are obtained using Understand for Java (http://www.scitools.com/) software.

TABLE 1
SOFTWARE METRICS

S.No	Metric	Definition	Source
1.	Coupling Between Objects (CBO)	CBO for a class is a count of the number of other classes to which it is coupled and vice versa	[15, 18]
2.	Number of Children (NOC)	The NOC is the number of immediate subclasses of a class in a hierarchy.	[15, 18]
3.	Number of Methods per Class (NOM)	NOM is the total number of methods defined in the class.	[16, 18]
4.	Number of Attributes per Class (NOA)	NOA is the total number of attributes/variables defined in the class	[16, 18]
5.	Number of Instance Method (NIM)	It is the total number of Instance Methods.	[17]
6.	Number of Instance Variable (NIV)	It measures the relations of a class with other objects of the program.	[17]
7.	Number of Local Methods (NLM)	Number of local (not inherited) methods.	[19]
8.	Response For a Class (RFC)	A count of methods implemented within a class and the number of methods accessible to an object class due to inheritance.	[15, 18]
9.	Number of Local Default Visibility Methods (NLDM)	Number of local default visibility methods.	[19]
10.	Number of Private methods (NPRM)	Number of local (not inherited) private methods.	[19]
11.	Number of Protected Methods (NPROM)	Number of local protected methods.	[19]
12.	Number of Public Methods (NPM)	Number of local (not inherited) public methods.	[17]
13.	Number Of Lines (NL)	Number of all lines.	[19]
14.	Blank Lines Of Code (BLOC)	Number of blank lines of code.	[19]
15.	Source Lines Of Code (SLOC)	The number of lines that contain source code.	[19]
16.	Lines of Declarative Code (LDC)	Number of lines containing declarative source code.	[19]
17.	Lines of Executable Code (LEC)	Number of lines containing executable source code.	[19]
18.	Lines of Comment (LC)	Number of lines containing comment.	[19]
19.	Statement Count (SC)	Total number of declarative and executable statements.	[19]
20.	Depth of Inheritance (DIT)	The depth of a class within the inheritance hierarchy is the maximum number of steps from the class node to the root of the tree and is measured by the number of ancestor classes.	[15, 18]
21.	Lack of Cohesion (LC OM)	For each data field in a class, the percentage of the methods in the class using that data field; the percentages are averaged the subtracted from 100%.	[15, 18]
22.	Weighted Methods Per Class (WMC)	The WMC is a count of the sum of complexities of all methods in a class.	[15, 18]

3.2 Dependent Variable

Change in any software is necessary to keep it updated and useful. Estimation of change prone classes of a software could help in better enhancements to a software at lower costs [10]. The dependent variable in our study is change proneness. Change proneness of a class is the likelihood of change after the software has been delivered [10]. Change is the modification made in a class in terms of SLOC added, deleted or modified.

4. EMPIRICAL DATA COLLECTION

In this section we give introduction to our data sources and provide a description of our data collection method.

We examined two open source software written in java language. The source code of these open source software is available at http://sourceforge.net/. The change in a software is accounted for, by gathering changes in all classes of that particular software. Two versions of a particular software were taken and the change in classes present in both the versions was analyzed. The software we studied were a) Frinika, a complete music workstation software for Linux, Windows, Mac OSX and other operating systems running Java 1.5 b) Freemind, a mind mapper and hierarchical editor. The details of the software are provided in Table 2. It shows the version, release date, SLOC and number of classes for each of the software analyzed in this study.

TABLE 2
SOFTWARE DETAILS

Name	**Version**	**Release Date**	**SLOC**	**No. of Classes**
Frinika	0.2.0	04-07-2006	49,532 SLOC	539
	0.6.0	22-10-2009	1,13,093 SLOC	1273
Freemind	0.9.0 RC1	21-12-2008	83,495 SLOC	920
	0.9.0 RC7	06-03-2010	85,766 SLOC	955

4.1 Data Collection Method

The procedure adopted for collecting OO metrics and changes between multiple versions of a class is as follows. It is similar to the one followed by Malhotra et al [10]. First, we compute all OO metrics specified in Table 1 for initial versions (Frinika-0.2.0 and FreeMind 0.9.0RC1) with the help of Understand for Java (http://www.sitools.com/) software. These metrics are only computed for all classes of the software. The second step is to prepare comparable versions of the software. For this, common classes are extracted from both initial and recent versions of Frinikaand FreeMind software. These classes are compared line by line with the help of Understand for Java to compute number of SLOC ADDED, DELETED and MODIFIED. TOTAL CHANGE for class is calculated by counting each ADDED or DELETED line as one SLOC change and each MODIFIED line as two SLOC changei.e. one deletion followed by one addition [8,10]. Another binary variable ALTER is defined which has the value 'yes' if TOTAL CHANGE > 0 or 'no' otherwise [10]. The change statistics (ADDED SLOC, DELETED SLOC, MODIFIED SLOC, TOTAL CHANGE and ALTER) and OO metrics are put together to generate data points. Malhotraet al. [10] provide a detailed and step by step procedure for data collection.

ForFrinika, out of 539 classes of Frinika 0.2.0, 80 classes were those which were deleted i.e. not present in 0.6.0. Only 248 classes were present in both the versions leading to 248 data points for Frininka. After data analysis, it was found that out of 248 data points 127 (51%) classes were altered while 121 classes were not changed in version 0.6.0. For FreeMind, all 920 classes were present in both the versions but library classes and unknown classes were discarded. This resulted in 657 data points.Out of 657 data points, 68 (10%) classes were found to be changed while the other 589 classes were not changed in 0.9.0 RC7 version.

5. RESEARCH METHODOLOGY

This section explains the various Machine Leaning (ML) methods we use for our analysis. We have used the default setting of WEKA tool.

5.1 Correlation based Feature Selection

While using a machine learning method, it is essential to diagnose and observe representative set of features [19]. Features in our case would be various OO metrics. A well established feature set is the one in which all features are highly correlated with the dependent variable, but these features should not be dependent on one another [20]. The dependent variable for our study is change prone classes.

Correlation based Feature Selection (CFS) aids in identifying repetitive and unwanted noisy features which should be removed. It has been established that classification results are more accurate if we use a reduced feature set as compared to a complete feature set [20]. CFS uses a correlation method among variables to identify good as well as noisy features. With the help of CFS we reduce execution time and improve predictive accuracy [21].

5.2 Decision Table

A Decision Table (DT) is a hierarchical structure of data in which complex data entries are present at higher levels of the table [22]. These complex entries are broken down into simpler entries with the help of additional attributes to form the table hierarchy. Attribute selection for break down of the table is measured by performing cross validation of all combinations of present attributes [6]. The subset which gives the best result is chosen.

5.3 Bayesian Network

A Bayesian Network (BN) is an interconnected network of nodes, where each node represents a random variable and all directed edges connecting these nodes represent probabilistic dependencies among nodes [6]. A BN should not contain a cycle. BN helps in computing joint probability distribution among a set of random variables. The structure of a BN represents conditional independence assertions [23]. BN can easily handle incomplete data sets and also allows us to investigate casual relationships.

5.4 Adaptive Boosting

Adaptive Boosting (AB) is a technique in which we combine many weak performance classifiers to obtain a strong performing classifier. This is done by building subsequent models which are based on gaining expertise on instances which are incorrectly classified by previous models [6]. It uses voting and assigns weights to its instances on the basis of performance [6]. The better performing model is given more weight. AB is sensitive to noisy data and outliers.

6. ANALYSIS RESULTS

In this section we describe the analysis performed for inter-project validation for change proneness using object oriented metrics. We employed DT, BN and AB machine learning methods for our analysis. The models predicted were applied to all the data points collected from the software. The following measures are used to evaluate the performance of each predicted model:

1. The sensitivity and specificity of the model are calculated to estimate correctness of the model. The percentage of classes correctly predicted to be change prone is known as sensitivity of the model [10]. The percentage of classes predicted not to be change prone is called specificity of the model. Ideally, both sensitivity and specificity should be high to predict change-prone and non change-prone classes [10].

2. Receiver operating Characteristic (ROC) analysis: The performance of the outputs, of the predicted models were assessed using ROC analysis. The ROC curve is defined as a plot of sensitivity on the y-coordinate versus 1-specificity on the x-coordinate [5]. An optimal choice of the cutoff point (that maximizes both sensitivity and specificity) can be chosen from the ROC curve.Area Under the ROC Curve(AUC) is a combined measure of sensitivity and specificity [5,10]. In order to compute theaccuracy of the predicted models, we use the area under the ROC curve.

3. Ten-fold cross validation of all prediction models is performed which analyzes the accuracy of the model. The data set is randomly divided into ten subsets. A number of iterations are performed by using one of the ten subsets as the test set while the other nine subsets are used for training. Therefore, we get the change proneness prediction for all the ten subsets [24].

6.1 Descriptive Statistics

A table for each software, Frinika and FreeMindis presented in the following subsection which show Minimum (Min), Maximum (Max), Mean (Mean), Median (Med) and Standard Deviation (SD) for all metrics considered in this study. The following observations are made from Tables 3 and 4.

1. The size of class measured in terms of SLOC ranges from 3 to 1538 (Frinika) and 1 to 1513 (FreeMind).

2. The mean values of NOC (Frinika -- 0.53, FreeMind -- 0.32) and DIT (Finika -- 3.32, FreeMind -- 1.98) are poor for both the projects which shows that there are only few subclasses and inheritance is not much used in these projects; similar results have been shown by others [17, 25-26].

3. The LCOM measure has high values (upto 100) in both data sets.

TABLE 3
DESCRIPTIVE STATISTICS FOR FRINIKA SOFTWARE

Metric	Min	Max	Mean	Med	SD
CBO	0	32	7.62	2	61.78
NOC	0	12	0.53	0	16.33
NOM	0	20	0.97	0	17.62
NOA	0	28	2.54	1	25.49
NIM	0	79	13.84	3	109.98
NIV	0	135	9.25	2	75.07
NLM	0	79	14.81	3	117.53
RFC	0	120	23.94	5	189.02
NLDM	0	9	0.79	0	16.95
NPRM	0	15	0.75	0	16.88
NPROM	0	10	0.36	0	16.03
NPM	0	73	12.9	3	102.75
NL	4	2047	255.94	57	2018.52
BLOC	0	264	44.64	10	351.89
SLOC	3	1538	167.15	40	1319.53
LDC	2	365	47.35	12	373.38
LEC	0	1054	85.72	18	679.89
LC	0	381	49.36	5	390.68
SC	2	1270	120.38	28	951.91
DIT	1	4	3.32	2	30.35
LCOM	0	100	93.64	50	735.27
WMC	0	163	28.25	6	222.92

TABLE 4
DESCRIPTIVE STATISTICS FOR FREEMIND SOFTWARE

Metric	Min	Max	Mean	Med	SD
CBO	0	148	4.96	3	7.87
NOC	0	18	0.32	0	1.49
NOM	0	63	0.3	0	2.63
NOA	0	37	0.91	0	2.81
NIM	0	129	7.39	4	12.66
NIV	0	105	2.38	1	5.94
NLM	0	129	7.69	4	12.97
RFC	0	222	14.83	6	23.83
NLDM	0	21	0.33	0	1.22
NPRM	0	16	0.95	0	2.15
NPROM	0	23	0.47	0	1.75
NPM	0	116	5.95	3	11.06
NL	1	2829	109.07	46	217.97
BLOC	0	244	13.5	6	27.42
SLOC	1	1513	78.75	33	143.15

LDC	1	471	23.64	11	41.84
LEC	0	766	38.46	15	75.84
LC	0	1544	17.86	3	70.52
SC	1	1152	56.07	24	103.51
DIT	1	6	1.98	2	1.04
LCOM	0	100	34.88	33	35.97
WMC	0	250	14.26	7	26.69

TABLE 5
DESCRIPTIVE STATISTICS OF DEPENDENT VARIABLE

Software Name	Dependent variable	Min	Max	Mean	SD
Frinika	TOTAL CHANGE	0	1166	46.202	140.024
FreeMind	TOTAL CHANGE	0	94	1.516	8.358

Table 5 presents the descriptive statistics of Frinika and FreeMind software. It displays the TOTAL CHANGE, the dependent variable which was calculated (Section 4.1) for both the software with its Minimum (Min), Maximum (Max), Mean and Standard Deviation (SD).

6.2 *Result Analysis for FreeMind Software using Ten-Fold Cross Validation*

First we applied CFS on data points generated by FreeMind software to select metrics for predicting change proneness. According to Table 6, the metrics selected were CBO, NOM, NOA, RFC, NLDM, NPRM, NPROM, NPM, BLOC, LDC, LC AND SC. Table 7 shows the validation results on FreeMind software using ten-fold cross validation. DT in this case gives the best results with AUC of 0.733 and specificity and sensitivity of 68.3% and 69.1% respectively. The cut off point for DT is 0.097. AB also gives good results with AUC of 0.709 and specificity and sensitivity of 60.6% and 69.1% respectively. The cut off point for AB is 0.061. The results of BN are comparable. BN gives AUC of 0.673. The specificity, sensitivity and cut off point for BN is 63.7%, 63.2% and 0.075 respectively.

TABLE 6
METRICS SELECTED BY CO-RELATION BASED FEATURE SELECTION

Software Name	Metrics Selected
Frinika	CBO, RFC, NIV, BLOC, LDC
FreeMind	CBO, NOM, NOA, RFC, NLDM, NPRM, NPROM, NPM, BLOC, LDC, LC, SC

TABLE 7
FREEMIND VALIDATION RESULTS USING TEN-FOLD CROSS VALIDATION

Method	Specificity	Sensitivity	Cutoff Point	AUC
DT	68.3	69.1	0.097	0.733
BN	63.7	63.2	0.075	0.673
AB	60.6	69.1	0.061	0.709

6.3 *Result Analysis of model prediction using Frinika as Training Set*

For using Frinika as training set and FreeMind as test set, we first selected metrics that predict change proneness by applying CFS. CFS was applied on data points generated by Frinika software. The metrics selected after applying CFS on Frinika were CBO, RFC, NIV, BLOC and LDC (Table 6). After applying CFS on Frinika data set, we use it as training set and supply FreeMind data points as test data using WEKA tool. The validation results obtained are present in Table 8. Here AB gives the best results with AUC of 0.767 and8 specificity and sensitivity of 69.9% and 70.6% respectively. The cut off point for AB is 0.742. After AB, DT showed good results with AUC of 0.742. The specificity and sensitivity with DT was 62.3% and 77.9% respectively and the cut off point was 0.608. Alternatively, the results with BN are also good with AUC of 0.727, specificity and sensitivity of 63.0% and 72.1% respectively and cut off point of 0.653. 6.4 Discussion of Results

TABLE 8
VALIDATION RESULTS USING FRINIKA AS TRAINING SET

Method	**Specificity**	**Sensitivity**	**Cutoff Point**	**AUC**
DT	62.3	77.9	0.608	0.742
BN	63.0	72.1	0.653	0.727
AB	69.9	70.6	0.613	0.767

The data points generated by FreeMind were validated using a) ten-fold cross validation and b) by using Frinika data set as the training set. FreeMind validation results for predicting change prone classes using both the above mentioned approaches were comparable. All the three machine learning techniques DT, BN and AB gave competent results for predicting change proneness. While DT, BN and AB gave AUC value of 0.733, 0.673 and 0.709 respectively when we use ten-fold cross validation, the AUC values while using Frinika as the training set were 0.742 0.727 and 0.767 respectively for DT, BN and AB.These results indicate that we can use a training set of one software project on anothersoftware project effectively while applying prediction models for change proneness. Such a technique is called inter-project validation. We concluded successful inter-project validation as results using FreeMind's own training set in ten-fold cross validation and the results by using Frinika (a different software) as the training set were comparable and competent.Reuse of training set for other software projects helps in better utilization of limited resource like time, cost and effort. Computing training sets specific to each software is a laborious and time consuming process. Thus effective re-utilization of already computed data sets can be done by employing them as training sets while applying prediction models on new data sets (software). Figure 1 below shows the process of inter-project validation using FreeMind as test set and Frinika as training set.

Another result obtained is that CBO, RFC and LDC metrics are good indicators of change proneness. These metrics were selected in the feature sets of both the data sets, after applying CFS.Prediction of change prone classes of a software,help us in better planning and administrating resources during maintenance. Figure 2 below shows the ROC curves obtained while applying DT, BN and AB on FreeMind using ten-fold cross validation while Figure 3 shows ROC curves on FreeMind, when Frinika data set is used for training.

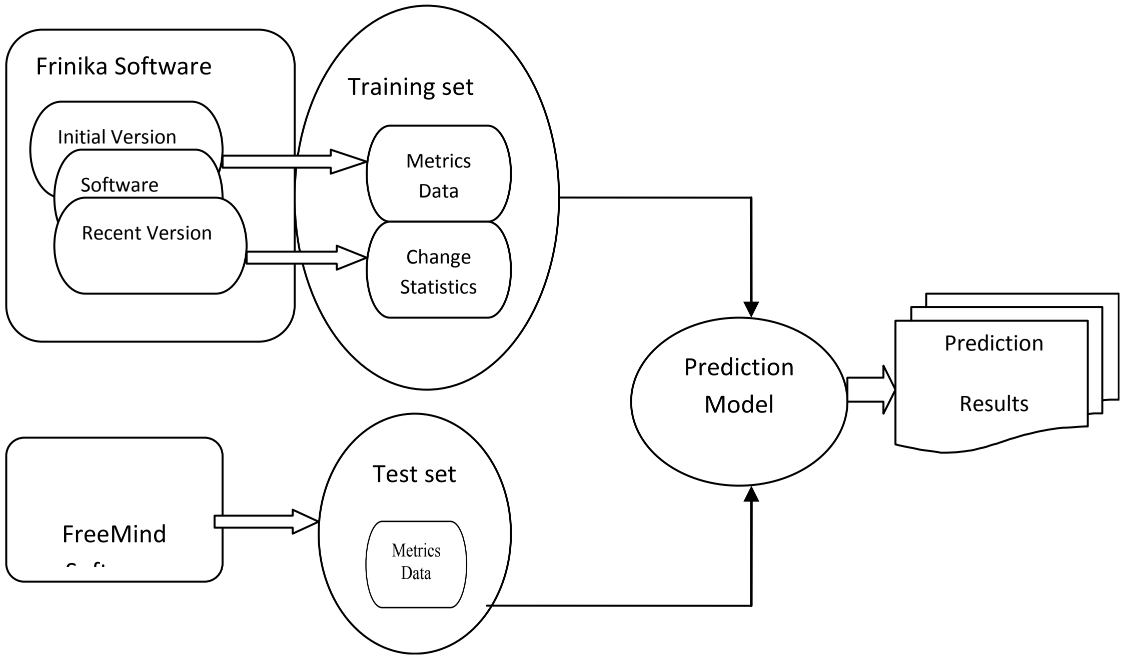

FIGURE : INTER PROJECT VALIDATION DIAGRAM

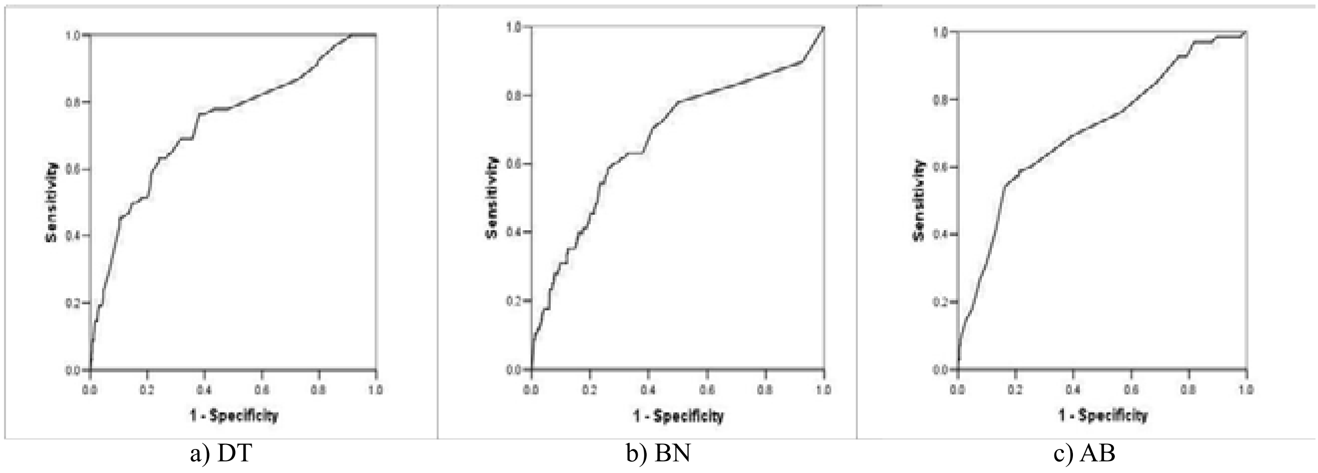

FIGURE 2 : ROC CURVE FOR FREEMIND SOFTWRAE USING TEN-FOLD CROSS VALIDATION a) DT b) BN c) AB

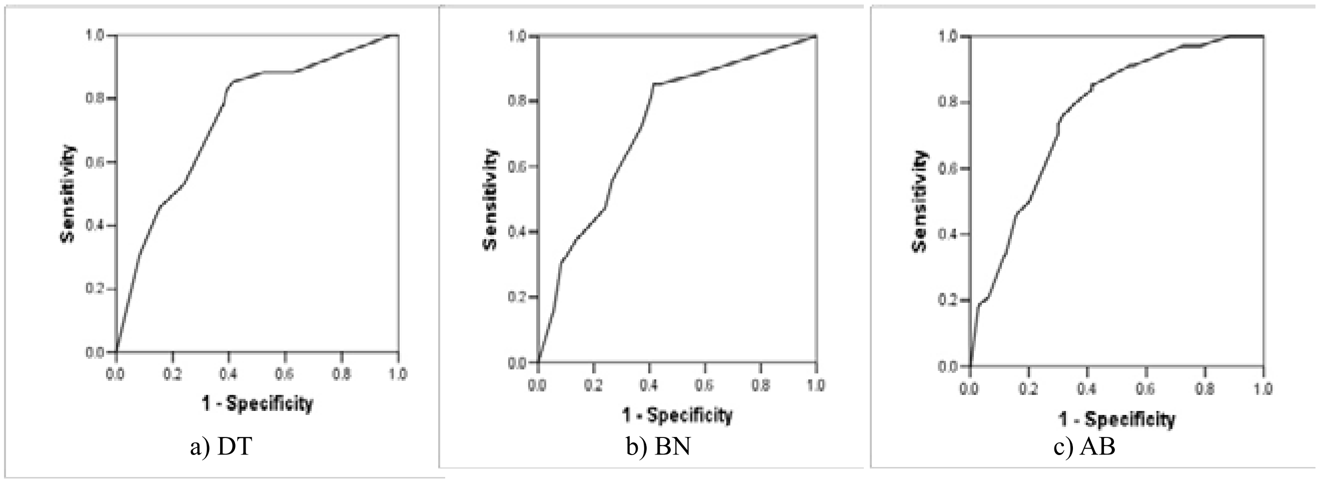

FIGURE 3 : ROC CURVE FOR FREEMIND SOFTWARE USING FRINIKA AS TRAINING SET a) DT b) BN c) AB

7. THREATS TO VALIDITY

In this section we discuss the various threats to validity: construct, external and internal which are prevalent in our study.

7.1 Construct Validity

The dependent and independent variables are measures of certain concepts. Construct validity is attributed to the degree of accuracy to which these concepts are measured with these variables [8]. Our independent variable, i.e. the various OO metrics (Table 1) have already been validated in previous studies[25, 27-28] as to whether they accurately represent the concepts like coupling, cohesion, inheritance etc. The values of various metrics are collected using Understand for Java tool. The tool does not specify the method followed for collection of these metrics for individual classes which would be a significant indicator of their construct validity. Change proneness, our dependent variable is collected manually by setting the ALTER variable in accordance with the change in the corresponding class. This eliminates construct validity threat with respect to the dependent variable. Software changes can be categorized as corrective, adaptive, perfective or preventive [8, 10, 29]. For our study, we did not classify the various kinds of changes as in [10].

7.2 External Validity

External validity is associated with the extent of generalization of results which can be demonstrated by comparative analysis on various data sets [8,10]. We cannot assure whether our results are universally applicable as we have not accounted for different programming languages and environments. This threat can only be minimized by conducting replicated studies with data sets having different characteristics.

7.3 Internal Validity

Internal validity is the extent to which we can summarize the casual effect of independent variable like coupling, cohesion, inheritance etc on the dependent variable i.e. change proneness [8]. In order to comment on casual effect we need to perform controlled experiments where a particular OO construct (like coupling) is varied and other constructs like cohesion and inheritance remain constant [2]. It is difficult to perform such an experiment and this threat exists in our study.

7. CONCLUSION AND FUTURE WORK

The goal of our research was to empirically examine inter project validation for predicting change prone classes using OO metrics. Based on the studies of data sets obtained from two open source software Frinika and FreeMind, we analyzed the performance of predicted models using ROC analysis. We applied machine learning methods to predict the effect of OO metrics on change proneness. We also took account of changes in the classes while predicting change proneness. Our main results are summarized as follows.

1. The metrics CBO, RFC and LDC are significant indicators of predicting change prone classes as in both the data sets, these metrics were chosen after applying feature subset selection method CFS. A good feature sets has features that are highly correlated with the dependent variable but are uncorrelated with each other [19]. CFS uses a correlation method among variables to identify good as well as noisy features.

2. The predicted results indicate that inter project validation of OO metrics for change proneness prediction is valid and yields competent results.While DT, BN and AB gave AUC value of 0.733, 0.673 and 0.709 respectively when we use ten-cross validation, the AUC values while using Frinika as the training set were 0.742 0.727 and 0.767 respectively for DT, BN and AB. These results indicate that we can use a training set of one software project on another software project effectively while applying prediction models for change proneness.While using ten-fold cross validation we use the training set of the same project which incorporates various software characteristics like domain, language etc of the particular software. But when we used Frinika as the training set and applied on FreeMind as the test set i.e. reuse a training set of a different project on another, the results were still comparable to that often-fold cross validation.This result helps research analysts and practitioners in reusing prediction models based on one software project on another project for prediction of change prone classes. This leads to effective and efficient utilization of resources and effort and time reduction as we do not need to compute training sets of specific software but can reuse an already existing set of a different software.

3. The results help us in effective utilization of constraint testing and maintenance resources. Change prone classes of a software need to be tested exhaustively during maintenance. Thus more testing resources can be assigned to these change prone classes for rigorous testing. Meticulous testing of change prone classes leads to a good quality software. Our results indicate that CBO, RFC and LDC are suggestive indicators of change prone classes. We can plan our limited testing resources in such a manner that they give competent good quality software products.

The results of our study are valid for object oriented medium systems. We plan to replicate our studies on data sets having different characteristics such as datasets with different programming languages and environments. We also plan to use other machine learning algorithms like genetic algorithms for inter-project analysis.

REFERENCES

[1] W. Li and S. Henry, “Object Oriented Metrics that Predict Maintainability,”*Journal of Systems and Software,* vol. 23, pp. 111-122,1993.

[2] L. Briand, J. Wust and H. Lounis,“Replicated Case Studies for Investigating Quality Factors in Object Oriented Designs.”*Empirical Software Engineering: An International Journal,* vol 6, pp. 11-58, 2001.

[3] 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,* vol. 22, no. 10, pp. 751-761, 1996.

[4] K.K. Aggarwal, Y. Singh, A. Kaur and R. Malhotra,“Empirical Analysis for Investigating the Effect of Object-Oriented Metrics on Fault Proneness: A Replicated Case Study.,”Software Process: Improvement and Practice, vol. 16, no.1, pp. 39-62, 2009.

[5] Y. Singh, A. Kaur and R. Malhotra. “Empirical Validation of Object-Oriented Metrics for Predicting Fault Proneness,” *Software Quality Journal,* vol. 18, no.1, pp. 3-35, 2010.

[6] I.H Witten and E. Frank, D*ata Mining : Practical Machine Learning Tools and Techniques,* 2nded, Elsevier, 2005.

[7] S. Watanbe, H. Kaiya and K. Kaijiri, “Adapting a Fault Prediction Model to Allow Inter Language Reuse,” *PROMISE 2008*, 2008.

[8] Y. Zhou, H. Leung and B. Xu,“Examining the Potentially Confounding Effect of Class Size on the Associations between Object Oriented Metrics and Change proneness,” *IEEE Transactions on Software Engineering,* 35, no. 5, 2009

[9] A.R. Sharafat and L. Tavildari, “Change Prediction in object Oriented Software Systems : A Probabilistic Approach,” *11th European Conference on Software Maintenance and Reengineering,* 2007.

[10] R. Malhotra and M. Khanna, “Investigation of Relationship between Object-oriented Metrics and Change Proneness,” *International Journal of Machine Learning and Cybernetics,* Springer ,2012

[11] A.R. Han, S. Jeon, D. Bae and J. Hong,“Behavioral Dependency Measurement for Change Proneness prediction in UML 2.0 Design Models,” *Computer Software and Applications 32nd Annual IEEE International,* 2008.

[12] M. D'Ambros, M. Lanza and R. Robbes, “On the Relationship Between Change Coupling and Software Defects,” *16th Working Conference on Reverse Engineering,* pp. 135-144, 2009.

[13] T. Zimmermann, N. Nagappan, H. Gall, E. Giger and B. Murphy, “Cross-project Defect Prediction A Large Scale Experiment on Data vs. Domain vs. Process”, in Proceedings ofthe 7th joint meeting of the European Software EngineeringConference and the ACM SIGSOFT International Symposiumon Foundations of Software Engineering. ACM, 2009, pp.91–100.

[14] H. Lu, Y. Zhou, B. Xu, H. Leung and L. Chen, “The ability of object-oriented metrics to predict change-proneness: a meta-analysis”,*Empirical Software Engineering Journal, vol.* 17, no.3, pp 200-242, 2012

[15] S.R. Chidamber and C.F. Kemerer,“A Metrics Suite for Object Oriented Design,” *IEEE Transactions of Software Engineering,* vol. 20, no.6, pp. 476-493, 1994.

[16] B.Henderson-sellers. Object Oriented Metrics. Measures of Complexity. Prentice Hall

[17] M. Lorenz and J. Kidd, “Object-Oriented Software Metrics,”*Prentice Hall Object-Oriented Series,* Englewood Cliffs, N.J., 1994.

[18] K.K. Aggarwal, Y. Singh, A. Kaur and R. Malhotra,“Empirical Study of object-oriented metrics”,*Journal of Object Technology,* 5, no.8, pp. 149-173, 2006.

[19] http://www.scitools.com/features/metrics.php

[20] M.A. Hall,“Correlation-based Feature Selection for Discrete and Numeric Class Machine Learning,” *Proceeding of the Seventeenth International Confrence on Machine Learning*,. pp. 359-366, 2000.

[21] K. Michalak and H. Kwasnicka,“Correlation-based Feature Selection Strategy in Neural Classification,” *Sixth International Conference on Intelligent Systems Design and Applications*, Vol 1, pp. 741-746.

[22] R. Kohavi and D. Sommerfield, “Targeting Business Users with Decision Table Classifiers,” *Proceedings of IEEE Symposiomon Information Visulaization*, 1998.

[23] F. Ruggeri , F. Faltin and R. Kenett,En*cyclopedia of Statistics in Quality & Reliability*, Wiley & Sons (2007).

[24] M. Stone,“Cross-validatory choice and assessment of statistical predictions,” *Journal of the Royal Statistical Society Series A*, Vol. 36, pp.111-14, 1974.

[25] L. Briand, J. Wust, J. Daly and D.V. Porter,“Exploring the Relationships between Design Measures and Software Quality in Object-oriented Systems,” *Journal of Systems and Software,* 51, no.3, pp. 245-273, 2000.

[26] S.R. Chidamber, D.P. Darcy and C.F. Kemerer., “Managerial Use of Metrics for Object-Oriented Software: An Exploratory Analysis,” *IEEE Transactions on Software Engineering*, 24, no.8, pp. 629-639, 1998.

[27] L. Briand, J. Daly, J. Wust, “A Unified Framework for Cohesion Measurement in Object Oriented Systems”, *Empirical Software Engineering Journal*, 3, no.1, pp 65-117, 1998.

[28] L. Briand, J. Daly, J. Wust, “A Unified Framework for Coupling Measurement in Object Oriented Systems”, *IEEE Transactions on Software Engineering*, 25, no.1, pp 91-121, 1999.

[29] K.K. Aggarwal, Y.Singh, *Software Engineering*, 3rd Edition, New Age International Publishers (2008).

ABOUT THE AUTHORS

Ruchika Malhotra. is an Assistant Professorat the Department of Software Engineering, Delhi Technological University (formerly Delhi College of Engineering), Delhi, India. She was an assistant professor at the University School of Information Technology, Guru Gobind Singh Indraprastha University, Delhi, India. She received her master’s and doctorate degree in software engineering from University School of Information Technology, Guru Gobind Singh Indraprastha University, Delhi, India. Her research interests are in software testing, improving software quality, statistical and adaptive prediction models, software metrics, neural nets modeling, and the definition and validation of software metrics. She has published more than 70 research papers in international journals and conferences. Malhotra can be contacted by email at ruchikamalhotra2004@gmail.com

Megha Khanna. is currently working as an Assistant Professor in Acharya Narendra Dev College, University of Delhi. She completed her Masters degree in Software Engineering (2010) from University School of Information Technology, Guru Gobind Singh University, India. She received her graduation degree in Computer Science (Hons.) (2007) from Acharya Narendra Dev College, Delhi University, India. Her research interests are in improving software quality, neural nets, artificial intelligence and the definition and validation of software metrics. She can be contacted by e-mail at meghakhanna86@gmail.com.