# Cross Project Change Prediction Using Open Source Projects

Ruchika Malhotra
Software Engineering Department
Delhi Technological
Delhi, India
Ruchikamalhotra2004@yahoo.com

Ankita Jain Bansal
Software Engineering Department
Delhi Technological University
Delhi, India
Ankita.bansal06@gmail.com

***Abstract*— Predicting the changes in the next release of software, during the early phases of software development is gaining wide importance. Such a prediction helps in allocating the resources appropriately and thus, reduces costs associated with software maintenance. But predicting the changes using the historical data (data of past releases) of the software is not always possible due to unavailability of data. Thus, it would be highly advantageous if we can train the model using the data from other projects rather than the same project. In this paper, we have performed cross project predictions using 12 datasets obtained from three open source Apache projects, Abdera, POI and Rave. In the study, cross project predictions include both the inter-project (different projects) and inter-version (different versions of same projects) predictions. For cross project predictions, we investigated whether the characteristics of the datasets are valuable for selecting the training set for a known testing set. We concluded that cross project predictions give high accuracy and the distributional characteristics of the datasets are extremely useful for selecting the appropriate training set. Besides this, within cross project predictions, we also examined the accuracy of inter-version predictions.**

***Keywords—Cross Project, Inter-version prediction, Change prediction, Object oriented paradigm, Machine learning, Metrics***

## I. INTRODUCTION

Change prediction deals with detecting change prone modules and thus, helps in allocating the resources (time, money and manpower) judiciously. Identification of change prone modules will bring some insights on the design of the software, such as it can indicate about the extent of coupling, cohesion, inheritance etc. used in the system. Thus, based on these insights, if required we can alter the design. There are various object oriented metrics proposed in literature ([1], [2], [3], [4], [5], [6], [7], [8]) that help in the prediction of change prone classes. Several empirical studies are done ([9],[10],[11],[12],[13],[14],[15],[6],[17]) to develop various change proneness models. But these studies have predicted the model by training it using the historical data from the same project. The predicted model was used to further identify change prone modules in the upcoming or future releases. However, prediction of change prone models using the training data from the same project is not always feasible as it might be possible that the adequate training data is not available or is not well collected ([18],[19],[20]).The possibility of non-availability of training data has led to the idea of performing cross project predictions. There have been few studies that have carried out cross project prediction for defect proneness ([18], [19], [20], [21], [22], [23]). Zimmermann, Nagappan and Gall [18] ran 622 cross-project defect predictions and found that only 21 predictions worked successfully (all Precision, Recall, and Accuracy are greater than 75%). The authors, Turhan, Menzies and Bener ([19], [23]) also conducted cross-company defect prediction and concluded that cross-company predictions increase the probability of defect detection at the cost of increasing false positive rate. Watanabe, Kaiya and Kaijiri [21] found that data characteristics of the datasets are important for cross-project defect prediction.

But to the best of our knowledge, the work of cross project predictions for change proneness has not been done yet. Hence, this has motivated us to conduct cross project predictions for change proneness. Cross project change prediction is defined as predicting the changes by training the models using the historical data of some other projects ([18], [21]). In other words, to predict changes in a project B, we train the models using the data of the other project A, and then test them on the project B. We can observe that training and testing are performed on the different projects, instead of using the same project. In this study, under cross project predictions, we have considered two categories: inter-versions predictions and inter-project predictions. Inter-version predictions refer to the predictions carried on two different versions of the same project. Whereas, inter-project predictions refer to the predictions carried on two different projects.

In this work, we tried to investigate the following two issues or research questions (RQ):

RQ1: How does the accuracy of the model differ when we train the model using the different versions of the same project

978-1-4799-3080-7/14/$31.00 ©2014 IEEE

(i.e. performing inter-version validation) or when we use different project (i.e. performing inter-project)?

RQ2: Are characteristics of data sets valuable for selecting suitable training data for cross-project defect prediction?

Aiming at the above questions, we conducted an experiment on 12 public datasets from 3 projects. All the three projects are Apache projects (Apache POI, Apache Rave and Apache Abdera) and we considered 4 latest releases for each project, thus making a total of 12 projects. We considered all the possible combinations (A, B) to perform cross project predictions for change proneness. We employed a machine learning technique, logitboost to construct the prediction models (to measure the performance of each cross project predications) with the help of the open source data mining tool WEKA. To determine the effectiveness of distributional characteristics in selecting the training data, we constructed a 'Classification and Regression Tree' (CRT) and generated various classification rules.

The different sections of the paper are: Following this section is the section 2 which explains the research background; the variables used in the study and various performance measures used. In next section, we explain our datasets along with the data collection process. Section 4 describes the basic four steps involved in this study. We discuss the results in section 5 and conclude the study in section 6.

## II. Research Background

In this section, we explain the independent and dependent variables used in this study. We also explain the various performance measures that are used to evaluate the performance of the models.

### A. Variables Used

We have used the famous object oriented metrics proposed by Chidamber and Kemerer [5] as independent variables. Along with the metrics proposed by Chidamber and Kemerer, one additional metrics is used, SLOC. Each of these metrics is defined below:

Coupling between objects (CBO): Number of other classes to which a class is coupled and vice versa

Response for a class (RFC): Number of methods in the class including the methods that are called by class's methods

Lack of cohesion (LCOM): For each data field in a class, the percentage of the methods in the class using that data field; the percentages are averaged and then subtracted from 100%

Number of children (NOC): Number of immediate subclasses of a class in a hierarchy

Depth of inheritance (DIT): Maximum number of steps from the class node to the root

Weighted methods per class (WMC): Count of sum of complexities of all methods in a class

Source lines of code (SLOC): Number of lines of code

Changes in software are inevitable and therefore, predicting the classes that are change prone in the future release of the software is highly beneficial. We have used dependent variable as change proneness which is defined as the probability of prediction of change in the next version of the software. This prediction is based on the changes in the current or present version. We have compared the classes of the two versions and measured changes in terms of number of lines of code added, deleted or modified and modified in the class of first (previous) version with respect to the class of the same name in the current version.

### B. Performance Measures

We have constructed a number of prediction models (for each possible train/test (A, B) combination) using a machine learning technique, logitboost. We evaluated each of these models using two performance measures explained in this section. Based on certain values of these performance measures, we have categorized each prediction as 'successful/valid' or 'not valid'.

There can be following categories of results:

- TNCP: number of classes correctly predicted to be 'not change prone'.
- FNCP: number of classes that are incorrectly predicted to be 'not change prone'
- FCP: number of classes that are incorrectly predicted to be 'change prone'
- TCP: number of classes that are correctly predicted to be 'change prone'

Precision: In terms of above categories, we define precision mathematically as TCP/ (TCP+FCP).The ideal value of precision is 1.0.

Area under the Receiver Operating Characteristics Curve (ROC): ROC curve is defined as a plot of sensitivity and 1-specificity on the y and x coordinates respectively ([24],[25]). We calculate sensitivity and specificity at different cut-off points between 0 and 1. We calculate area under the ROC curve which is a measure of overall performance of the predicted model. Higher the area under the curve, higher is the accuracy of correct prediction [26].

## III. Empirical Data Collection

We have used 4 releases each of 3 open source Apache projects, Abdera, POI and Rave. Each release is considered as a different dataset, thus making a total of 12 datasets. Apache Abdera (http://abdera.apache.org/) is an implementation of the Atom Syndication Format and Atom Publishing Protocol which are used for creating, editing, and publishing various web

sources. Apache POI (http://poi.apache.org/) is used for making and modifying different file formats based upon OOXML and Microsoft's OLE 2. We can read/ write Word, Excel and PowerPoint files using Java. It is used in text extraction applications such web spiders etc. Apache Rave (http://rave.apache.org/) provides an extensible platform for using, integrating and hosting various OpenSocial and W3C Widget related features, technologies and services. Rave is popularly used as engine for internet and intranet portals.

The releases that we have considered are as follows (1) for Abdera: Abdera 1.0, Abdera 1.1, Abdera 1.1.1 and Abdera 1.1.2, (2) for POI: POI 3.0, POI 3.6, POI 3.7 and POI 3.9, (3) for Rave: Rave 0.19, Rave 0.20.1, Rave 0.21.1 and Rave.22. The details of each of these releases which include the release year, total number of classes and the number of classes changed in each release are shown in table I. We have calculated change between every two successive versions. For example, change is calculated for the common classes between the releases Abdera 1.0 and Abdera 1.1. We briefly describe the change collection process as follows:

For collection of the data, we used a tool named Defect Collection and Reporting System (DCRS), developed in Java by the students of Delhi Technological University, Delhi, India. The tool is used to generate the change reports for open source software hosted at *Git repository*. The tool functions as follows: Two consecutive versions of software are taken and their source code is analysed. Using Git commands, we process the source code and retrieve Change-Logs. A Change Log contains all the information concerned with the modifications that have been incurred from time to time in the source code. The modification in the source code could be due to various reasons such as defect-fixing, function/feature enhancements etc. Each modification is stored as an individual change record in the Change log. An individual change record at the Git repository consists of: a) Change timestamp, b) Unique change identifier, c) Description of the change including the reason why the change has occurred, d) Source code files that are modified, along with the LOC changes for each modified file.

Finally, these Change records are mapped to source code Classes. Only java source code files are considered and other files such as xml and other resources are ignored. The change- report generated consists of the following fields:

- Name of the Java source file (Class name).
- A binary variable to indicate whether a class has been modified, i.e. 'yes' else 'no'.
- Corresponding data for Metrics for each class.

TABLE I. DETAILS OF THE PROJECTS

| Project | Written in / License | Releases | Release Date | No. of classes | No. of Classes Changed |
|---|---|---|---|---|---|
| Abdera | Java / Apache 2.0 | Abdera 1.0<br>Abdera 1.1<br>Abdera 1.1.1<br>Abdera 1.1.2 | May 2010<br>July 2010<br>Dec. 2010<br>Jan.2011 | 679<br>677<br>686<br>685 | 624<br>8<br>634<br>635 |
| POI | Java / Apache 2.0 | POI 3.0<br>POI 3.6<br>POI 3.7<br>POI 3.9 | July 2007<br>Dec.2009<br>Oct. 2010<br>Dec. 2012 | 1515<br>2088<br>2472<br>2786 | 1276<br>1988<br>2212<br>2706 |
| Rave | Java / Apache 2.0 | Rave 0.19<br>Rave 0.20.1<br>Rave 0.21.1<br>Rave 0.22 | Dec. 2012<br>Mar. 2013<br>Apr. 2013<br>May 2013 | 607<br>642<br>676<br>685 | 32<br>628<br>648<br>225 |

## IV. RESEARCH METHODOLOGY

Here, we explain the basic four steps involved in this study. The first step is the creation of train/test combination or generation of datapoints. This is followed by measuring distributional characteristics for each train/test combination. The third step is the construction or prediction of machine learning model to determine the accuracy of cross project predictions. Finally, we construct a Classification and Regression tree (CRT) to judge the suitability of characteristics of data sets for selecting the training data for cross-project defect prediction. All the steps are explained diagrammatically in figure 1.

### *A. Creation of Train/Test Combinations*

We have used 3 Apache projects (Apache POI, Apache rave, Apache Abdera), with four releases of each. A project with 4 releases is considered as 4 separate datasets and therefore, 3 projects will comprise of 12 datasets. To perform cross project validation, one dataset among the 12 datasets is taken as training set and some other dataset is taken as the testing set. We have considered the possible combinations of training and testing sets (A, B) if: 1) A and B belong to different projects or 2) A and B belong to same project, then B should be the later version than A [18]. In other words, we want the testing set to be the later version than the training version as it is illogical to perform testing on the former version and training on the later version. For example, we used Abdera 1.1.1 to predict the later version, Abdera 1.1.2 and not the former version, Abdera 1.1.

Therefore, the number of possible combinations with A,B,C…..N projects, having number of releases as a,b,c….n respectively:

$$P_2^T - \sum_{i=A}^{N} \frac{v_i * (v_i - 1)}{2}$$

Where, T= number of datasets = a+b+c….+n

$V_i$ = number of releases of $i_{th}$ project. For example, number of releases for A project is a.

Thus, based on the above formula, the total possible combinations in this study are 114.

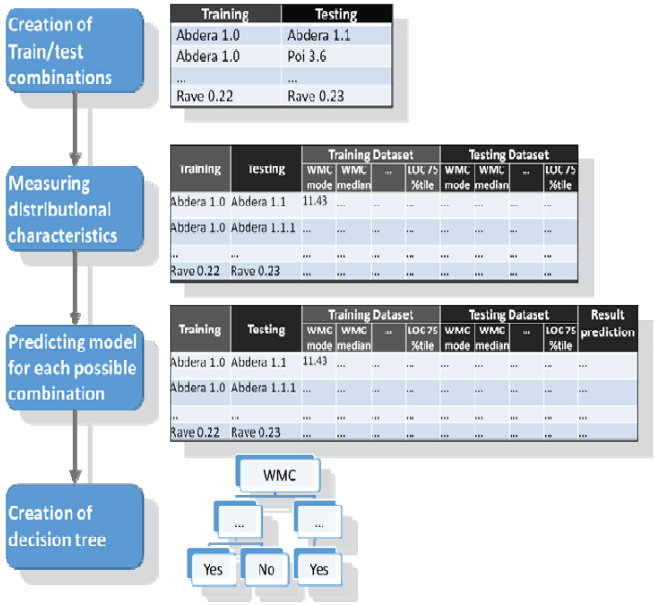


Fig. 1. Basic Research Methodology

## B. Measuring Distributional Characteristics

We have used three open source projects with 4 releases of each. For these 12 different datasets, we calculate various descriptive statistics which measure the distributional characteristics of each independent variable (number of independent variables are 7). We measure 11 distributional characteristics (indicators) for each variable. As discussed, we have 114 possible train-test combinations. We create 144 attributes for each combination (or datapoint) as follows:

1. We combine the distributional characteristics of each variable of training set to get 77 (7 independent variables *11indicators ) attributes
2. We combine the distributional characteristics of each variable of testing set to get 77 (7 independent variables *11indicators) attributes
3. 77 attributes of training set and 77 attributes of testing set are further combined to yield a total of 144 attributes.

The various indicators that we used to measure the distributional characteristics are Mode, Median, Mean (to describe the central tendency of attributes), standard deviation, variance, skewness, kurtosis, minimum, maximum, sum, first quartile and third quartile.

## C. Predicting Model for each Possible Prediction

We have used a machine learning algorithm, logitboost to build the model for each valid training-testing pair (A, B). We predicted the model using the dataset A and tested it on the dataset B. Thus, in this study, we have 114 model predictions. For each prediction, we say whether the prediction is valid or not valid based on the values of performance measures, precision and area under the ROC curve. If the value of area under the ROC curve is greater than 0.65 and precision is greater than 50%, we have considered the prediction as valid or successful. This we call as the 'result' of each prediction, which is discrete or binary in nature. In this study, for each prediction, we have defined a datapoint having three parts: 1) consists of distributional characteristics of training set, 2) consists of distributional characteristics of testing set, 3) the result of each prediction (i.e. valid or not valid). Thus, we have 114 datapoints, each having 145 attributes (77 number of attributes comprising of distributional characteristics of training set, 77 number of attributes comprising of distributional characteristics of testing set and 1 attribute comprising of the result of the prediction).

## D. Verify the relationship between Distributional Characteristics of data sets and Selection of Training Set

Our one of the objectives is to determine whether the distributional characteristics of training and testing set have an impact on the cross project predictions. The studies by Zimmermann, Nagappan & Gall [18] and Watanabe, Kaiya & Kaijiri [21] show that the distributional characteristics are essential for identifying successful cross-project defect prediction. In this work, we construct a decision tree on the dataset generated above (consisting of 114 datapoints) to check the accuracy of the decision tree in finding the valid cross-project predictions. The independent variables are the distributional characteristics of training and testing set (77+77=144 attributes) and the binary dependent variable is the result of each prediction (valid/not valid). The validation technique used is 10-cross validation. If the decision tree predicts the valid predictions with high accuracy, then we conclude that the cross project predictions have an impact on the distributional characteristics of the training and testing sets.

# V. Result Analysis

In this section, we address the two research questions stated in section 1 and intend to answer them

**5.1 How does the accuracy of the model differs when we train the model using the different versions of the same project (i.e. performing inter-version validation) or when we use different project (i.e. performing inter-project validation)?**

For each of the possible train/test combination (A, B), we predicted the machine learning model (logitboost) using dataset A and tested it on dataset B. We measured the performance in

terms of precision and area under the ROC curve. We are interested in knowing the number of successful cross project predictions. We refer a prediction as successful if the precision is more than 50% and area under the ROC curve is more than 0.65. We observed that out of a total of 114 predictions, 41 satisfy our criteria and thus are referred as successful predictions (i.e. the success rate is 36%). This is quite high when we compare it with the success rate obtained by [18]. Zimmermann, Nagappan & Gall [18] ran 622 cross-project predictions and found only 3.4% actually worked. Within the successful predictions, we want to compare the number of successful inter-project and inter-version predictions. Inter-version predictions are the predictions where the training and testing sets belong to the same project, but different versions. Whereas, in inter-project predictions, training and testing sets belong to different projects. It would seem that the different releases of same project will predict each other well, as they are developed under same environment, have similar characteristics etc. However, there is a difference of opinion in this issue by previous studies. Some of the previous studies ([28],[29],[30],[31]) have shown that different releases from the same project have predicted each other well. However, the latest study by He, Shu, Yang Li and Wang [32] had shown unexpected result, i.e. decreased prediction accuracy for the same (ranging from 0.32% to 4.67%).

In table II, we analyze the number of successful inter-project and inter-version predictions. If two datasets A and B belong to the same project, then the total possible train/test combinations (A,B : B should be later version than A), are 18 (6 combinations for each Abdera, POI and Rave). In other words, we say 18 is the total number of inter-version combinations. Out of 18 combinations, we observe that the number of successful predictions is 12, i.e. approximately 67% success rate. On the other hand, if two datasets A and B belong to the different project, then the total number of train/test combinations (A,B) are 96 (114-18). Thus, the number of inter-project combinations is 96. Out of 96, the number of successful predictions is 29, i.e. approximately 30% success rate. This is significantly lower than the success rate of inter-version predictions. Thus, in our study, we conclude that the inter-project predictions give higher accuracy than the inter-project predictions.

Table III shows the range of area under the ROC curve (AUC) for successful predictions. For Abdera as testing project, we see that using Abdera as training project (different version of same project), the range of AUC is 0.67 to 0.78. We can observe that the range of AUC when training set is Poi and Rave is also approximately same as when the training set is Abdera. This shows that the accuracy of the inter project and inter version predictions is approximately the same. Same observation is concluded when the testing project is Poi and Rave.

Table II. NO. OF SUCCESSFUL PREDICTIONS

| TRAIN/TEST | Abdera | POI | Rave | Total |
|---|---|---|---|---|
| Abdera | **5** | 5 | 5 | 15 |
| POI | 10 | **4** | 0 | 14 |
| Rave | 9 | 0 | **3** | 12 |
| Total | 24 | 9 | 8 | **41** |

Table III. AUC OF SUCCESSFUL PREDICTIONS

| Testing Project | Training Project | Min. AUC | Max. AUC |
|---|---|---|---|
| Abdera | Abdera | 0.67 | 0.78 |
| | Poi | 0.67 | 0.71 |
| | Rave | 0.67 | 0.71 |
| Poi | Abdera | 0.66 | 0.72 |
| | Poi | 0.67 | 0.69 |
| | Rave | - | - |
| Rave | Abdera | 0.65 | 0.72 |
| | Poi | - | - |
| | Rave | 0.66 | 0.68 |

### 5.2 Are characteristics of data sets valuable for selecting suitable training data for cross-project defect prediction?

We generated a total of 114 possible predictions and the number of successful predictions is 41. To determine whether the characteristics of the datasets help in selecting the suitable training set, we applied decision tree (CRT) on the dataset of 114 datapoints having 145 attributes (144 attributes consisting of distributional characteristics of training and testing sets + 1 attribute for the result of prediction). CRT trained from the train/test instances consisted of 15 nodes, out of which 8 are leaf nodes. The leaf nodes will contain the value of the result of prediction (dependent variable). We observed 3 nodes resulted in the value 'yes', from which we generated the classification rules. These classification rules can be used to select a suitable training set for a given test set. We also performed 10-cross validation on this dataset and obtained precision as 85.6 % and the area under the ROC curve as 0.98. Taking into consideration success rate for inter-project predictions (67%) and the success rate for inter-version predictions (30%), we can say that the performance of CRT is very high. Thus, we can conclude that the distributional characteristics of the datasets play an important role in selecting the suitable training set for a given test set.

## VI. CONCLUSION

Many a times, in various commercial organizations, the availability of historical data becomes questionable as it is unavailable or is insufficient. Generally, the models for change prediction are learned using the historical data of some project. Then, using this trained/learned model, we predict the changes

in the upcoming releases of that project. But non-availability of historical data poses a difficult scenario. In such cases, we may use the data from some other project rather than the same project. This is referred to as cross project prediction, i.e. training and testing the model on different projects. Careful selection of training data from other projects is very important. Thus, cross project prediction is gaining wide importance for projects with insufficient or no data to build prediction models. Despite its importance, to the best of our knowledge, no work has been done in the field of cross project change prediction. In this paper, we have used 4 releases each of 3 open source projects, Apache Abdera, POI and Rave. We have total 12 datasets (4*3) and we made all the possible combinations (A, B) of training/testing set. We calculated the total number of possible combinations is 114 and thus, we ran 114 cross project predictions using a machine learning technique, logitboost. Following are the main conclusions that can be drawn from the study:

1. We have considered the successful predictions as those cross project predictions where precision is more than 0.5 and area under the ROC curve is more than 0.65. Using this criterion, we found a success rate of 36%.
2. We conducted both cross project (different projects) and inter version (different versions of same projects) predictions. Among the successful number of predictions, we found that the success rate of inter-version (different versions of the same project) is 67% and the success rate of cross project predictions (different projects) is 30%. From this, we concluded that inter-version predictions are more accurate than the inter-project predictions.
3. We constructed a CRT decision tree to judge the suitability of characteristics of the datasets in selecting the training set for a given test set. We found high values of precision, recall and area under the ROC curve. This shows that selection of training set depends on the distributional characteristics of the datasets. For a given test set, we can select an appropriate training set using the classification rules (that produced a 'yes') of CRT.

In our future work, we would like to experiment our methodology on more number of projects. This would help us to generalize the results. Besides this, we would try to find out if cross-project change prediction is symmetrical. In other words, we will see if project A predicts project B well, does it imply that vica-versa is also true (i.e., if project B also predicts project A well). Finally, we would propose a well defined training data selection technique. For a given testing set, we would propose a technique to find its suitable training set.

## REFERENCES


[1] L. Briand, W. Daly and J. Wust, "Unified framework for cohesion measurement in object-oriented systems," Empirical Software Engineering, vol. *3*, pp. 65–117, 1998

[2] L. Briand, W. Daly and J. Wust, "A unified framework for coupling measurement in object-oriented systems," IEEE Transactions on Software Engineering, vol. 25, pp. 91–121, 1999.

[3] J. Bieman and B. Kang, "Cohesion and reuse in an object-oriented system", in Proceedings of the ACM Symposium on Software Reusability (SSR'94), pp. 259–262.

[4] M. Cartwright and M. Shepperd, "An empirical investigation of an object-oriented software system," IEEE Transactions on Software Engineering, vol. 26, pp. 786-796, 1999.

[5] S. Chidamber and C. Kemerer, "A Metrics Suite for Object-Oriented Design," IEEE Trans. Software Eng., vol. 20, pp. 476-493, 1994.

[6] W. Li and W. Henry, "Object-oriented Metrics that Predict Maintainability," Journal of Software and Systems, vol. 23, pp. 111-122, 1993.

[7] M. Lorenz and J. Kidd, Object-Oriented Software Metrics, Prentice-Hall, 1994.

[8] J. Bansiya, and C. Davis, "A Hierarchical Model for Object-Oriented Design Quality Assessment," IEEE Trans. Software Eng., vol. 28, pp. 4-17, 2002.

[9] M. Askari and R. Holt, "Information Theoretic Evaluation of Change Prediction Models for Large-Scale Software," in ACM: Proceedings of international workshop on Mining software repositories, pp.126-132, 2006.

[10] A.R. Han, S.U. Jeon, D.H. Bae and J.E. Hong, "Behavioural Dependency Measurement for Change-proneness Prediction in UML 2.0 Design Models," Annual IEEE International Computer Software and Applications Conference, 2008.

[11] A.G. Koru and J. Tian, "Comparing High-Change Modules and Modules with the Highest Measurement Values in Two Large-Scale Open-Source Products. IEEE Transactions on Software Engineering," vol. 31, pp. 625–642, 2005.

[12] A. Gu¨nes, Koru and H. Liu, "Identifying and characterizing change-prone classes in two large-scale open-source products," The Journal of Systems and Software, vol. 80, pp. 63–73, 2007.

[13] D. Romano and M. Pinzger, "Using Source Code Metrics to Predict Change-Prone Java Interfaces," Tech. Rep. ISSN 1872-5392.

[14] M. Lindvall, "Are large C++ classes change-prone? An empirical investigation," Software Pract Ex., vol. 28, pp. 1551−1558, 1998.

[15] E. Arisholm, L.C. Briand and A. Føyen, "Dynamic Coupling Measurement for Object-Oriented Software," IEEE Transactions on Software Engineering, vol. 30, pp. 491-506, 2004.

[16] 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, vol. 35, pp. 607-623, 2009.

[17] H. Lu, Y. Zhou, B. Xu, H. Leung and L. Chen, "The ability of object-oriented metrics to predict change-proneness: a meta-analysis," Empir Software Eng., vol. 17, pp. 200-242, 2011.

[18] T. Zimmermann, N. Nagappan and H. Gall, "Cross-project defect prediction: a large scale experiment on data vs. domain vs. process," In: Proceedings of the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering, pp. 91–100, 2009.

[19] B. Turhan, T. Menzies and A. Bener, "On the relative value of cross-company and within_company data for defect prediction," Empir. Softw. Eng., vol. **14**, pp. 540–578, 2009.

[20] M. Jureczko and D. Spinellis, "Using object-oriented design metrics to predict software defects," In: Proceedings of the 5th International Conference on Dependability of Computer Systems, pp. 69–81, 2010.

[21] S. Watanabe, H. Kaiya and K. Kaijiri, "Adapting a fault prediction model to allow inter language reuse," In: Proceedings of the International Workshop on Predictive Models in Software Engineering, pp. 19–24, 2008.

[22] N. Nagappan, T. Ball, and A. Zeller, "Mining metrics to predict component failures," In: Proceedings of the 28th International Conference on Software Engineering, pp. 452–461, 2006.

[23] B. Turhan, A. Bener and T. Menzies, "Regularities in learning defect predictors," In: The 11th International Conference on Product Focused Software Development and Process Improvement, pp. 116–130, 2010.

[24] K. El Emam, S. Benlarbi, N. Goel and S. Rai, "A validation of object-oriented metrics," NRC Tech. rep. ERB-1063, 1999.

[25] Y. Singh, A. Kaur and R. Malhotra, "Empirical validation of object-oriented metrics for predicting fault proneness," Softw Qual J, vol. 18, pp. 3–35, 2010.

[26] T. Fawcett, "An introduction to ROC analysis*,"* Pattern Recognition Letters, vol. 27, pp. 861–874, 2006.

[27] R. Malhotra and M. Khanna, "Investigation of relationship between object-oriented metrics and change proneness," Int. J. Mach. Learn. & Cyber., vol. 4, pp. 273-286, 2013.

[28] A. Tosun, B. Turhan and A. Bener, "Practical considerations in deploying AI for defect prediction: a case study within the Turkish telecommunication industry," In: Proceedings of the 5th International Conference on Predictor Models in Software Engineering, pp. 1–9, 2009.

[29] T.J. Ostrand, E.J. Weyuker and R.M. Bell, "Predicting the location and number of faults in large software systems," IEEE Trans. Softw. Eng., vol. 31, pp. 340–355, 2005.

[30] E.J. Weyuker, T.J. Ostrand and R.M. Bell, "Comparing the effectiveness of several modeling methods for fault prediction," Empir. Softw. Eng., vol. 15, pp. 277–295, 2009.

[31] E.J. Weyuker, T.J. Ostrand and R.M. Bell, "Do too many cooks spoil the broth? Using the number of developers to enhance defect prediction models," Empir. Softw. Eng., vol. 13, pp. 539–559, 2008.

[32] Z. He, F. Shu, Y. Yang, M. Li and Q. Wang, "An investigation on the feasibility of cross-project defect prediction," Autom Softw Eng, vol. 19, pp. 167–199, 2012.