Journal of Software Engineering and Applications, **2011, 4, 639-645**
doi:10.4236/jsea.2011.411075 Published Online November 2011 (http://www.SciRP.org/journal/jsea)

Defect Prediction Leads to High Quality Product

Naheed Azeem, Shazia Usmani

Department of Computer Science, Federal Urdu University of Arts, Science and Technology, Sindh, Pakistan.
Email: {naheedazeem, shaziausmani}@fuuast.edu.pk

Received August 12th, 2011; revised September 25th, 2011; accepted November 5th, 2011.

ABSTRACT

Defect prediction is relatively a new research area of software quality assurance. A project team always aims to produce a quality product with zero or few defects. Quality of a product is correlated with the number of defects as well as it is limited by time and by money. So, defect prediction is very important in the field of software quality and software reliability. This paper gives you a vivid description about software defect prediction. It describes the key areas of software defect prediction practice, and highlights some key open issues for the future.

Keywords*: *Defect Prediction, Software Quality

1. Introduction

Software life cycle is a human activity, so it is impossible to prevent the injection of defects but it is possible to produce the software with few defects. To deliver a defect free software it is imperative to predict and fix the defects as many as possible before the product delivers to the customer.

Finding and fixing the defects after delivery usually consumes a large portion of the project budget. Therefore, defect prediction before delivery can contribute signifycantly to the success of project in terms of quality and cost.

The aim of this research is to explore the different issues and problems in the area of defect prediction as well as provide the solutions to improve the product quality via defect prediction mechanism.

In this survey report several research issues, formulated as questions, need to be addressed to understand the problems of defect prediction mechanism.

Research questions:

- How machine learning algorithms and data mining techniques can be prove more effective in defect extraction from repository?
- What kinds of software metrics are good indicators of defects?
- To what extent the number of defects injected in the software product can be reduced?
- How can we easily identify and localized the software defects?
- What kind of software repository could represent the required information?
- What methods or procedures are better to opt for defect identification and localization?
- How can we reduce the probability of false alarm?
- How good predictor is in finding actual defective modules?
- Is there a type of defect prediction model that provides a good fit to defect-prediction across multiple releases and in many organizations?
- To what extent can we use other project data to predict defects for a software system and is there any possibility to transfer prediction models from one project to another?

The rest of this paper is outlined as follows. We begin by providing background and descriptions in section 2. The Issues and problems encountered by the defect prediction are elaborated in section 3. The methods and approaches used to tackle issues are illustrated in section 4.In section 5 some future research areas are presented. Finally, we finish with conclusion.

2. Background and Descriptions

A ***software defect*** is an error, flaw, mistake, failure, or fault in a computer program or system that produces an incorrect or unexpected result, or causes it to behave in unintended ways.

Software defects are expensive in terms of quality and cost. Moreover, the cost of capturing and correcting defects is one of the most expensive software development activities. It will not be possible to eliminate all defects but it is possible to minimize the number of defects and their severe impact on the projects. To do this a defect

management process needs to be implemented that focuses on improving software quality via decreasing the defect density. A little investment in defect management process can yield significant returns.

Software Defect Prediction

The most discussed problem is software defect prediction in the field of software quality and software reliability. As Boehm observed finding and fixing a problem after delivery is 100 times more expensive than fixing it during requirement and design phase. Additionally software projects spend 40 to 50 percent of their efforts in avoidable rework.

In summarizing the major research trends for defect prediction of software products include:

Software Metrics

Software metric is a measure of some characteristic or attribute of software module. Since Software metrics are quantitative methods and have proved so powerful in defect prediction. The essence of software quality engineering is to investigate the relationship between different metrics and end-product quality.

Software metrics can be classified into three categories: product metrics, process metrics, and project metrics. Product metrics describe the characteristics of the product such as size, complexity, design features, performance, and quality level. Process metrics can be used to improve software development and maintenance. The project parameters such as the number of developers and their skill levels, the schedule, the size, and the organization structure certainly affect the quality of the product.

Defect Identification

Identifying and locating defects in software projects is a difficult task. Further, estimating the density of defects is more difficult. So, the software project team is fully focused on finding and fixing all the defects.

Defective and Defect-free modules

Accuracy of defect prediction techniques is determined by correctly finding the defective parts of a software product without giving any false alarm. Giving high false alarm rate means developers and testers wasting their time in inspecting and testing defect free modules. On the other hand, predictions of defective modules as defect free modules would cause more expensive in terms of quality and cost.

Data Mining and Machine Learning Techniques

Machine learning models and Data mining techniques can be applied on the software repositories to extract the defects of a software product. Common algorithms include decision tree learning, Naive Bayesian classification and neural networks.

3. Issues and Problems

3.1. Problem with Selecting the Right Set of Metrics

- Studies based on accuracy of defect prediction model focused on either project metrics or product metrics but not the combined impact of both [1].
- It is strongly believed that software size has a relationship with software quality but there is a lack of evidence that shows size metrics as a good indicator of defects [2].
- Defect prediction model did not conclude that either change metrics or code metrics were better for defect removal [3].
- Managers rely on complexity metrics to allocate QA resources effectively, but complexity metrics fail to predict critical binaries of a complex system [4].
- The effectiveness of SSM as defect predictor in OO software needs to be established [5].

3.2. Problem in Reducing False Alarm

- It is difficult to find the optimum threshold value that makes the difference between defective and non defective modules [6].
- Static attributes are mainly used in decreasing false alarms but they do not provide the enough information to significantly reduce the rate of false alarms [7].
- The influence of refactoring on defects prediction process do not provide any conclusive result i.e. either refactoring or non refactoring related features leads to high quality defect prediction model [8].
- Implementation of Defect management process in multi-site software development organization is difficult and more challenging [9].
- Relationship between change coupling and software defect were unknown [10].

3.3. Defect Identification Issue

- Most of the defect prediction models do not utilize customer profile and system characteristics to predict the customer reported defects in order to improve the defect prediction mechanism [11].
- Traditional capture recapture model do not estimate the number of defects in post inspection phase and also rely on expert inspectors [12].
- Simulation approach and queuing theory used to model defect removal process did not consider the utilization of developers [13].
- Only the number of defects cannot provide enough information to support the software quality activities [14].

- Traditionally, some defect prediction models are used to identify the number of defects in a multi-version system but they are not platform and language independent [15].

3.4. Problem with Extraction of Defects from Repository

- Extraction of defects from software bug repository accurately is not done without a good data mining model [16].
- There is a need of good data mining model to predict the software defects from a bug repository [17].
- Most of the machines learning algorithms are not capable of extracting defects from the database that store continuous features [18].
- Prior research on defect prediction fails to fully utilize the defect data and defect repair time estimation requires mathematical assumptions [19].
- Software prediction model only works well when enough amount of data is available in software repository within the organization to initially fed the model [20].

4. Approaches and Methodologies

4.1. Metrics as a Predictor of Software Defects

Only the few authors claimed that project metrics are also helpful to improve the quality of software product. Wahyudin and Schatten studied the relationship of project metrics to the potential growth of defects and determined the combined impact of project metrics and product metrics on defect prediction .Two step predictor pro- cess was proposed in [1]. First find out the Pearson rank correlation among the strongest correlated predictors with the dependent predictor (variable) and then use the stepwise linear regression and backward elimination to exclude the insignificant predictors to form a reliability growth model. Result revealed that project metrics has strong correlation with the potential growth of defects between release and combined effect of project and product metrics were result in better prediction model.

Line of code (LOC) is one of the simplest and widely used metric. H. Zhang analyzed the. Two public defect datasets to prove the relationship of software size with software quality. The "ranking ability" of LOC can be actually modeled by a Weibull distribution function. By using defect density values calculated from a small percentage of the largest modules, LOC's ability to predict the number of defects can be improved. Also using typical classification techniques, defective components based on LOC are able to predict. Results showed that LOC can be a useful indicator of software quality, and useful to build defect prediction models using LOC [2].

In [3] another comparative analysis was done to check the effectiveness of change metrics over code metrics for defect prediction. Three different models one for change metrics, one for code metrics and one for both change and code metrics, using three machine algorithms i.e. logistic regression, Naive Bayes and decision tree (J48). Cost analysis was also performed to evaluate the cost associated with the prediction errors of these three models. Result showed that change metrics are significantly better indicator for defect prediction model than static code attributes.

Zimmermann studied that complexity metrics are failed to provide better result in defect prediction when it comes to critical binaries. A dependency graph of windows 2003 is build. For each node (binary) on dependency graph, network measures are computed. Several code metrics as a control set are applied to all the binaries [4].

Fenton *et al*. criticized that there is no relationship between complexity and defects as well as with the size. In spite of critique, most of the studies used SSM as indicator of defects in procedural paradigm as well as in OO software. The role of software science metrics (SSM) in defect prediction of object oriented (OO) software had been studied. Binary and numeric classification models available in WEKA are applied on dataset with class level data. The models are first applied using all the metrics available in the dataset and then removing SSM from the input and the accuracies and error values of all the models are observed. Effectiveness of SSM is measured at model level by comparing accuracies and Mean absolute error of models with and without SSM [5].

4.2. Software Defect Reduction

Performance of defect prediction mechanism is determined by the probability of defect detection and probability of false alarms. Reduction in the false alarm is done by a two-dimensional ROC analysis. The author changed the decision threshold on Naïve Bayes and observed the changes in prediction performance measures. Using decision threshold optimization on Naïve Bayes classifier, probability of false alarm (pf) rate has decreased, while Balance rate has increased and the probability of detection (pd) rates remained the same. These results were also validated using paired t-test [6]. 10 repository metrics were extracted from CVS revision system of eclipse project and classified the source files as defective if it contained one or more defects and non defected if it contained zero defect. Naïve Bayes used to extract the additional metrics from repository to increase the input data. Mann-Whitney U test was done on data to test the statical significance of using different techniques. Results showed that repository metrics give better insight to software product and hence able to lower pf rate as compared to using only static code attributes [7].

Earlier studies have addressed evolutionary activities areas such as refactoring based on history information. It was also investigated the influence of refactoring on defects prediction process. Approach discussed in [8] provide conclusive result that refactoring leads to high quality defect prediction model. Authors identify the refactoring features using evolution data extracted from versioning system. Machine learning algorithm WEKA is used to generate the section model made up of same number of defected and non defected files, and then the statistical analysis is performed on these models to evaluate the hypothesis.

In a multi-site organization when development is distributed, hundreds of people are working on a project; establish a defect prediction mechanism is too difficult. A GQM (Goal Question metric) method is adopted and a combination of three quality metrics was used [9].

Ambros and Lanza studied the relationship between coupling and software defects. They create a source code model and a history model and train them with the necessary data. The correlation of change coupling with the number of software defects, major defects and with severe defects is measured by using Spearman correlation coefficient on three large java systems [10] as shown in **Figure 1**.

4.3. Identifying and Locating Defects

Most of the researchers have focused on predicting the number of defects to improve the defect prediction mechanism but only the numbers of defects is not enough to get sufficient information that support quality activities.

Raaschou and Rainer raise this issue and build a model that based on customer reported defects. Exposure model takes detailed customer profile, reported defects and version data as input. Number of new defects at low level can be expressed as the product of the defected versions at initial release and isolated effect of five exposure factors. These factors influence the number of defects found and can be aggregated as needed to higher levels [11].

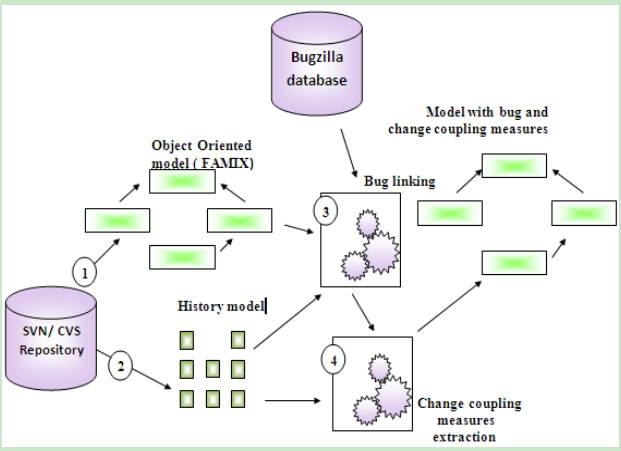

Figure 1. Creating a model with bug and change coupling information [10].

Bucholz investigated that static capture recapture model did not capture the defects when it is in post release phase. They extend the static capture recapture model. In dynamic capture recapture model for initial release, 30-40 size of capture recapture is fixed and for the next release heuristic algorithm is used to calculate the next capture recapture size. Identify the number of defects in each release and find out the duplicates by matching new reported defects to the previous defects existing in defect database. Calculate the total number of predicted defects by using Peterson estimator [12]. Fan and Xiaohu considered the developer data. Defect detection process was done by an algorithm which generates non homogenous poisonous process. Classified the new defects according to their severity and then assigned to the developers. Update the status of defects and fixed defects were removed from the queue [13].

Hong and Baik proposed a new approach for predicting the distribution of defects and their types based on project characteristics as can be seen in **Figure 3**. Determine the factors that affect software attributes being estimated then perform behavior analysis on them. Gather all the defect data and build a model. The model for defect prediction was built by using curve fitting method and regression analysis. After statistical modeling and regression analysis validates and refines the model. [14].

Kastro build a model that worked on different version of software and it is language and platform independent (**Figure 2**). In [15] Metric data include CVS level data,

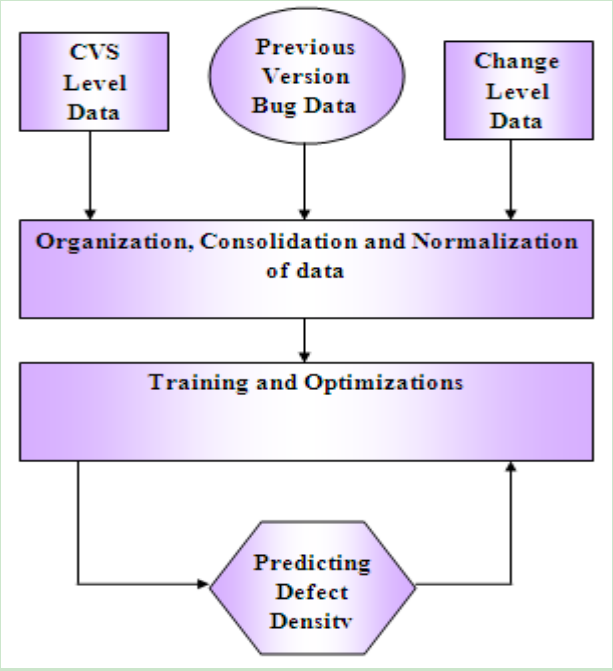

Figure 2. Proposed defect prediction model [15].

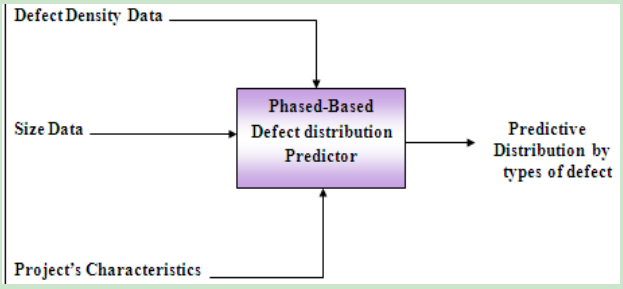

Figure 3 .Overview of the proposed model [14].

change level data and previous version data was collected. Collected data was organized, consolidate and normalized. Built a model by using multilayer perception with neural networks and trained the model with normalized data.

4.4. Data Mining and Software Repositories

Software repositories have lots of information that is useful in assessing software quality. Data mining techniques and machine learning algorithms can be applied on these repositories to extract the useful information.

Tosun and Bener applied AI technique in predicting defects. They extract static code attributes at functional level from the source code and then store the defect data. They construct and calibrate the defect prediction model using AI algorithm [16]. In [17] a two step data mining model is proposed to predict software bug estimation. In first step, a weighted similarity model is used to match the summary and description of new bug from the previous bug in the bug repository. In the second step calculate the duration of all the bugs and the average is calculated.

The technique used in [18] is entropy based splitting criteria and minimum description stopping criteria (decide when to stop discretization).The binary discretization was always selecting the best cut point and was applied recursively (**Table 1**). The authors investigated the effect of discretization on defect prediction models.

Hewett proposed a model that aid help in software testing and estimate defect repair time. Empirical approach employed data mining technique that increased the utilization of defect data in prediction of defect repair time to support testing and defect management. They used four

Table 1. Summary of the results [18].

Classifier	Software Modules	Correctly Classified Instances	
		Before Discretization %	After Discretization %
Naïve Bayes	121	85.124	95.562
	101	82.178	85.148
J48	121	90.082	92.562
	101	83.168	85.148

data mining algorithms based on three different approaches *i.e.* decision tree learner, Naïve Bayes classifier and neural network approach used to build a mode [19].

Zimmermann findings are in the case of when there is not enough historical data to train the model. In that scenario Zimmermann used cross project data to build and train the model. Relative measures such as code churn (added, deleted, and changed lines), domain metrics and process metrics extracted from development process of one project are used to build prediction model for another project, based on logistic regression. Their research result as can be seen in **Figure 4**. Accuracy, precision and recall are used to assess the model [20].

5. Future Work and Open Issues

Future work in this area should:

- Establish an improved method for predicting software quality via identifying the defect density of fault prone modules and improve the rate of false alarm [6,8,12].
- Different machine learning algorithm and data mining techniques are used to improve the defect prediction accuracy [16,18].
- Extracting automatically key information from the data repositories that are more relevant for defect prediction [7] and trying right set of metrics that influence the success of cross project predictions [19, 20].

6. Conclusions

Software defect prediction is the process of locating defective modules in software. To produce high quality software, the final product should have as few defects as possible. Early detection of software defects could lead to reduced development costs and rework effort and more reliable software. So, the study of the defect prediction is important to achieve software quality.

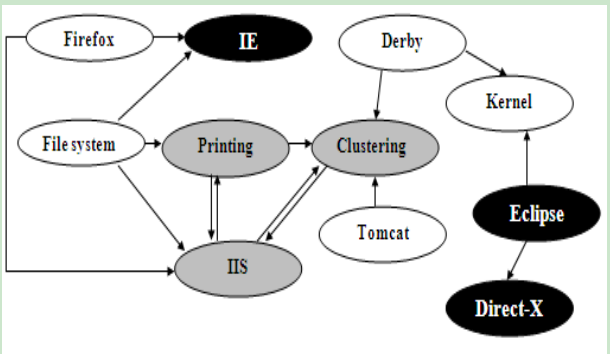

Figure 4. Results from 622 cross-project defect predictions. For example, Firefox data can predict IE. The color tells whether a project predicts other projects (white), can be predicted (black), or both (gray) [20].

Therefore our study aims to provide useful insight on the defect prediction approaches to aid project team in making quality product.

The findings of this research confirm observations made by other researchers that:

- History metrics extracted from repository helps in reducing false alarm as well as increasing the rate of probability of detection in open source software and LOC has positive relationship with software defects.
- Refactoring has great impact on software quality improvement as well as on building high quality defect prediction model.
- GQM method with combinations of quality metrics proved better to support the defect prediction process in multi-site organizations.
- The number of defects alone cannot be sufficient information to provide the basis for planning quality assurance activities and assessing them during execution. That is, for project management to be improved, we need to predict other possible information about software quality such as in-process defects, their types, and developer and customer profiles and so on.
- Change coupling is correlated with the number of software defects, major defects and with severe defects.
- Integration of discretization method with classification algorithm improves the defect prediction accuracy by transforming the continuous features into discrete features.
- Data mining techniques are useful in prediction of defect repair time, software bug estimation more accurately, and predicting the number of defects in multi-version environment that is language and platform independent.
- Cross project defect predictors build accurate prediction model when significant factors are evaluated and quantified.
- Network measures on dependency graph predict defect for critical binaries more accurately than complexity metrics.
- Use of software science metrics is ineffective for defect prediction and classification of defect prone modules in object oriented software.
- The comparative study on product metrics and process metrics concluded that overall change metrics were effectively better than code metrics as well as project metrics with combination of product metrics is more effective in defect prediction.

REFERENCES

[1] D. Wahyudin, A. Schatten, D. Winkler, A. M. Tjoa and S. Biffl, "Defect Prediction Using Combined Product and Project Metrics: A Case Study from the Open Source "Apache" MyFaces Project Family," *Proceedings of the* 2008 34*th Euromicro Conference Software Engineering and Advanced Applications*, 2008, pp. 207-215.

[2] H. Zhang, "An Investigation of the Relationships between Lines of Code and Defects," 2009 *IEEE International Conference on Software Maintenance*, pp. 274-283. doi:10.1109/ICSM.2009.5306304

[3] R. Moser, W. Pedrycz and G. Succi, "A Comparative Analysis of the Efficiency of Change Metrics and Static Code Attributes for Defect Prediction," *In Proceedings of the International Conference on Software Engineering* (*ICSE*'08), Leipzig, pp. 181-190.

[4] T. Zimmermann and N. Nagappan, "Predicting Defects using Network Analysis on Dependency Graphs," *In Proceedings of the International Conference on Software Engineering* (ICSE'08), Leipzig, Germany, pp. 531-540.

[5] Z. A. Rana, S. Shamail and M. M. Awais, "Ineffectiveness of Use of Software Science Metrics as Predictors of Defects in Object Oriented Software," *Proceedings of the* 2009 *WRI World Congress on Software Engineering*, Vol. 04, 2009, pp. 3-7. doi:10.1109/WCSE.2009.92

[6] A. Tosun and A. Bener, "Reducing False Alarms in Software Defect Prediction by Decision Threshold Optimization," *Third International Symposiumm on Empirical Software Engineering and Measurement*, 2009 IEEE, pp. 477-480. doi:10.1109/ESEM.2009.5316006

[7] B. Caglayan, A. Bener and S. Koch, "Merits of Using Repository Metrics in Defect Prediction for Open Source Projects," 2009 *ICSE Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development*, pp. 31-36.

[8] J. Ratzinger, T. Sigmund and H. C. Gall, "On the Relation of Refactoring and Software Defects," *In Proceedings of the International Workshop on Mining Software Repositories* (MSR '08), Leipzig, Germany, pp. 35-38.

[9] K. Korhonen and O. Salo, "Exploring Quality Metrics to Support Defect Management Process in a Multi-Site Organization—A Case Study," 19*th International Symposium on Software Reliability Engineering*(IEEE '08), pp. 213-218.

[10] M. D'Ambros, M. Lanza and R. Robbes, "On the Relationship between Change Coupling and Software Defects," *Proceedings of the* 2009 16*th Working Conference on Reverse Engineering*, IEEE 2009, pp.135-144.

[11] K. Raaschou and A. Rainer, "Exposure Model for Prediction of Number of Customer Reported Defects," *Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, (ACM '08), pp. 306-308.

[12] R. Bucholz and P. A. Laplante, "A Dynamic Capture-Recapture Model for Software Defect Prediction," *Innovations in Systems and Software Engineering* (2009), Springer London, pp. 265-270.

[13] W. Fan, Y. Xiaohu, Z. Xiaochun and C. Lu, "Simulation of the Defect Removal Process with Queuing Theory,"

3*rd International Symposium on Empirical Software Engineering and Measurement*, 2009, pp. 473-476.

[14] Y. Hong, J. Baik, I. Y. Ko and H. J. Choi, "A Value-Added Predictive Defect Type Distribution Model based on Project Characteristics," *Seventh IEEE/ACIS International Conference on Computer and Information Science*, 2008, pp. 469-474.

[15] Y. Kastro and A. B. Bener, "A Defect Prediction Method for Software Versioning," *Software Quality Journal*, Springer Netherlands, Vol. 16, 2008, pp. 543-562.

[16] A. Tosun, B. Turhan and A. Bener, "Practical Considerations in Deploying AI for Defect Prediction: A Case Study within the Turkish Telecommunication Industry," *Proceedings of the* 5*th International Conference on Predictor Models in Software Engineering*, 2009, pp. 24-25.

[17] N. K. Nagwani and S. Verma, "Predictive Data Mining Model for Software Bug Estimation Using Average Weighted Similarity," *IEEE* 2*nd International Advance Computing Conference*, 2010, pp. 373-378.

[18] P. Singh and S. Verma, "An Investigation of the Effect of Discretization on Defect Prediction Using Static Measures," *IEEE International Conference on Advances in Computing, Control, and Telecommunication Technologies*, 2009, pp. 837-839. doi:10.1109/ACT.2009.212

[19] R. Hewett, "Mining Software Defect Data to Support Software Testing Management," *Applied Intelligence*, Springer Netherlands, 2009.

[20] 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," *European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (*ESEC/FSE*), Amsterdam, 2009, pp. 91-100.

