Generalize Approach for Building Defect Prediction Model

Arun Singh, Member, IEEE and Rajesh Singh
Department of Computer Science
BSA College of Engineering & Technology
Mathura, India
aruncsmt10@bsacet.org, rajesh.singh@bsacet.org

***Abstract*— It is well known that defects are very much crucial for any type of software project. They increase the development time and cost, and degrade the quality and performance. Early identification of defective software modules is much beneficial for developing good quality software at low cost. An effective defect prediction model is needed to predict these defective modules during the project development. There are many defect prediction models are exist and most of them are constructed using publicly available defect data.**

In this paper we argued that, these defect prediction models are not very much efficient when the different software development organizations used them to predict the defects during their own software development. We also propose a generalized approach for constructing a defect prediction model. This approach is very much helpful for such organizations or individuals who want to construct the defect prediction model using their own software development defect data or also publicly available defect data.

Keywords—Defect Prediction; Defect Data; Software Quality; Defect Prediction Model; Data Mining.

I. INTRODUCTION

Quality have very much important for the software products. Only those software are admissible which contain good quality according to their users and customers [1] [17]. The functional capability, performance and accuracy of the software product are also depending on its quality. Every software development organizations want to develop the good quality software products but the remaining defects are one of the biggest contributors for degrading the quality and also responsible for various system failures. To preserve the quality, it is much necessary to eliminate these defects from the software. Defect correction or defect prediction tasks help to eliminate these defects. When we apply the defect correction process then we find that it is very expensive task [2] and take too long time [3] but despite of all, there is no surety to elimination of all defects [4]. However, defect prediction is a dexterous approach for early recognition of defects during software development.

The efficient defect prediction approach introduces various guidelines for identification of defective software modules and efficient allocation of testing resources. Once software development team got the knowledge about defective modules, they can easily find appropriate solutions and save the testing time. Overall, efficient defect prediction approach save the development cost and time and also helps to improve the software quality, reliability, and performance.

To perform an efficient defect prediction, an effective prediction model is required. The essential things for development of defect prediction model are prediction methodology and historical software development defect data. Data mining provide various efficient prediction methodologies [5] but in this paper our main rumination is, appropriate selection of historical software development defect data.

The historical software development defect data are very much useful for the investigation of software quality, risk evaluation and development efficiency for present and feature developments of software projects. These software defect data have many other benefits but one of the most valuable benefits is that they are used for defect prediction.

The defect prediction model is constructed through the various historical software development defect data stored in various data repository in the form of various data matrices. The defect data stored in some repositories are publicly available [6-8] and can be used for various research purposes. There are various studies [9-13] used these data to construct defect prediction model. The various different software development organizations can use to these defect prediction models for predicting the defects during their software development projects.

Zimmermann et al. [15] build the defect prediction models using Firefox and Internet Explorer (IE) development data. The prediction model built using Firefox development data can predict the defects in the development process of IE but model developed using IE development data is not capable to predict defects in the development process of Firefox whenever the most of the features of both products are similar but development process and organization is different.

In addition, Koru et al. stated [9] that defect prediction models will change from one development site to another according to specific defect patterns. Thus, the software product is being developed by any organization, if they use defect prediction model that is built using development data of other organization might not work well.

In this paper, our argument is that, the only sometime the prediction models built using publicly available defect data seems good, otherwise mostly they work inefficiently, when the organizations or individuals use them for their specific purpose. Because the efficient defect prediction is not only

depends on effective prediction methodology but also very much influenced by the nature of historical data used to build the prediction model. The prediction model gets the knowledge about the class distribution of sweep net of defects exists in historical software development by interpreting the historical defect data and then applies this knowledge to predicting the defects during present or future software development. The software development process is differs for different organization, therefore the defect data collected during development have different nature for different organization. Thus, the defect prediction models that are built using publicly available defect data, are contain only the knowledge of defect classes for historical software development of the same organization from where the historical defect data is collected, just because of, the defect prediction model is not properly applicable for software developments of different organizations.

This paper is organized as, section 2 discuss various factors due to them the software development process differs for different organizations, section 3 discuss that what should actual defect data needed to build the defection prediction model, section 4 discuss a generalized approach for development of defect prediction model, section 5 discuss the experimental analysis, section 6 discuss threats for validity of our experiment, section 7 concludes the this paper.

II. FACTORS AFFECT THE SOFTWARE DEVELOPMENT

As we already discussed that the development process of the software product is different for one organization to other, whether they are developing similar types of products. There are some following variables; due to the software development varies from one organization to other.

A. Organizational Infrastructure

Every software development organizations have different infrastructure. The infrastructure of any organization is not directly affects the development of software. It affects the development team and also the coordination and communication of team members. It also defines the working environment for development team members. Frederick strongly stated in his book [16] that the structure of organization is strongly affecting the quality of product.

B. Technology

The development technologies refer to the tools and techniques used in development of software products such as cost estimation tools, designing tools, various algorithms, operating systems, programming language etc. The development of software products depend the technologies are used, thus the different technologies leads to different development process such as if a software product is developed using JAVA then object oriented aspect is also become considerable.

C. Peoples

There are two types of peoples related with the development of the software products; first one who develop the products and second who use the products. The development of the software is highly depends on development skills of software development team members and the user background and user's knowledge about the product. Generally, the software product users may or may not different but the development team members are always different for different organizations.

D. Quality Level

A software product satisfying the user requirements by its performance means the software product has good quality level [1] [17]. If users are different for one to other organizations then the development process of software also differ whenever they developing similar type of product. Always good quality software is expected to be developed, but actuality is that, the organization develop a software product according to, where the software being used, such as the software used in projection of the satellite are highly reliable and developed with zero defects. Thus the different software development organizations develop the software product with different level of quality.

E. Defect Patterns

The software development is the human driven process so it cannot be complete without appearing the defects. The defects are widely classified as computational, Data Value, Interface, Initialization, Control/Logic defects classes [18] [19]. Most probable, the ratio of arising one type of defects during development at one organization differs from other organization. So they needed different approach to predict them.

F. Cost and Time

The development of software is highly dependent on cost and time. The sufficient cost and time of project make the software development smooth and successful but insufficient cost and time cause various problems with product outcomes [20] [21]. The different organizations have different cost and time limitations for the development of the software.

G. Scope

The scope has very much important for development of the software project and must be clearly defined for successful development of the software [22][23]. The different organizations define different scope for their software products.

H. Goals and Objectives

Only clearly defined goals and objectives can help a software project to meet the user requirements. The project goal and objective determines the required information to complete the development of the project [23]. The different organizations develop their software product with different goal and objectives.

III. APPRAISEMENT OF DEFECT DATA

The organizational data warehouse is a historical data repository. There are huge amount of historical defect data stored in this data repository. Data mining apply various metamorphose for pruning these defect data to extract the appropriate datasets which are required to build the defect prediction model. This data pruning is known as data preprocessing and includes various tasks [24] such as data cleaning, data integration, data reduction, and data transformation, but before applying the pruning, it is

necessary to know that what exactly data is required to build the defect prediction model. Berry & Linoff define some questions and their answers in his book [25] are help us to know the accurate data that are required to build the defect prediction model. Here we discuss these questions and answers in brief.

A. What is Available?

The data repository contains the large amount of data that are regularly updated by appending new data, but the historical data is never changed. Data in the warehouse has already been cleaned and verified and brought together from multiple sources. The data warehouse contains various types of organizational data such as sales, production, software development, marketing, customer and finance etc. related data.

B. How Much Data is Enough?

It depend on particular prediction methodology used, the complexity of the data, and the relative frequency of expected outcomes.

C. How Much History is Required?

The past software development data is used to build the defect prediction model for predicting the defects in present and feature software development. But the question is, how far in the past should this data come from? The time-period of technology changes is most considerable. If the data from too far in the past and during this period, the lot of changes may take place in the development technologies, the data may not be useful.

D. How Many Variables?

The good selection of variable is depends upon the experience of software practitioner. The selection of the data variables is depends on the prediction methodology. Some ignored variables may turn out the predictive outcomes so take it carefully.

E. What Must the Data Contain?

The purpose of defect prediction is to classify the software modules in to faulty or non-faulty class. So data must contain the examples of all possible outcomes that can help to identify the faulty or non-faulty modules. These data examples should contain the all types of defect values, whereby the defect prediction model build using these data become capable to identify all types of defective modules.

IV. Proposed Concept for Development of Defect Prediction Model

Turhan et al. [14] argued in his comparative study that the defect prediction models built using organizational personal development defect data work more efficiently for that organization, in compare to models built using publicly available defect data or other organization data. They also stated that the less amount organizational personal defect data are required to build an efficient defect prediction model and these data can be collected in few time. Thus, only those organizations should use the publicly available data that are developing the software at first time and no personal historical defect data are available. Koru et al. [9] suggested to software development organizations that, they should build defect prediction models using their own historical development defect data and try to make available as soon as possible, and then update them continuously.

We realize that, if organizations try to build the defect prediction model then they need to follow an efficient approach to build it. Thus, we discuss a generalized approach that helps the organizations to build defect prediction model using their personal defect data or publicly available defect data. The development task of defect prediction model can be categorized into five sections.

A. Data Re-assessment, shown in figure 1 from step 1 to 4 of flowchart.

B. Data Division, shown in figure 1 from step 5 to 8 of flowchart.

C. Model Building, shown in figure 1, step 7 and 9 of flowchart.

D. Model Finalization, shown in figure 1, step 6, 8, 10, and 11 of flowchart.

E. Apply the Model, shown in figure 1, step 12 of flowchart.

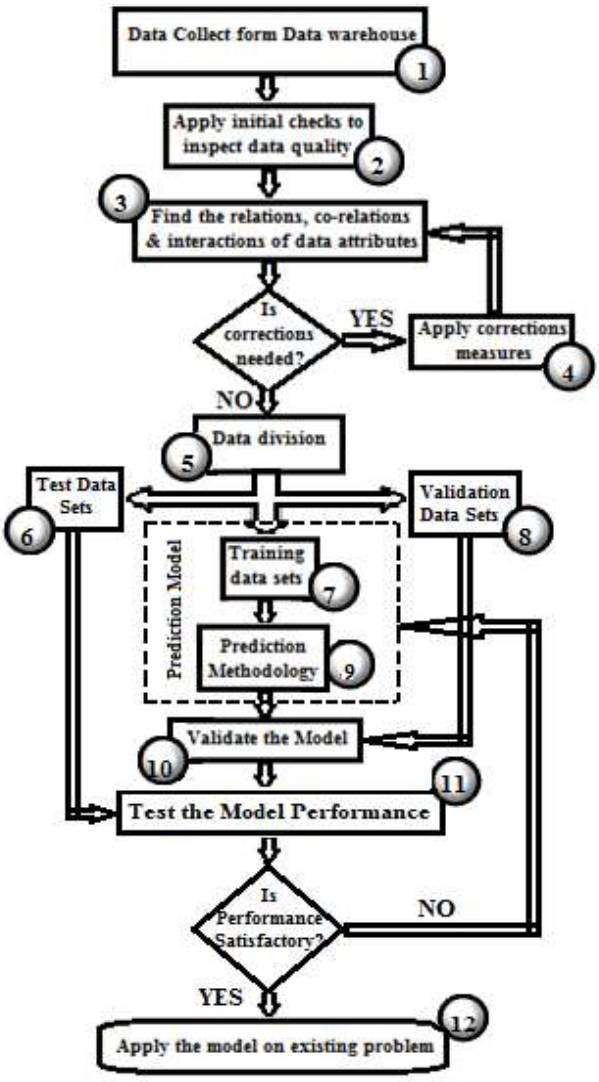

Fig. 1. Flowchart of Development Process of Defect Prediction Model.

A. Data Re-assessment

The task data preprocessing improve the accuracy, completeness, consistency, timeliness, believability, and interpretability of the historical defect data [24] and provides the knowledge of exact requirements of defect data to build the prediction model. This knowledge enables us to appropriate collection of defect data from historical data repository [25]. In spite of, we investigate the data relativity with the purpose of building effective defect prediction model.

1) Collect the Data: First of all, collected the preprocessed data from data warehouse and store this data into a place where from the data can be easily accessed.

2) Check the Data Quality: As we know the data preprocessing includes various tasks that improve the data quality level as needed to build prediction model, such as reduce the data at the required level and clean it appropriately by pruning unnecessary data [24]. But in spite of, no further any misconceptions take place; we apply the initial checks to inspect that the data quality is really matching the required level. These initial checks include the various tasks, such as identifying that, is data contains missing attribute values or noisy data, is data contains enough attribute that are required to build the model because more than enough and less than enough attribute degrade the predictive performance of the model [10][26][27] etc.

3) Evaluate the Data Attributes: It is not only enough that the datasets contain all attributes required to build prediction model but also it is very necessary to identify that, how these data attributes are relevance to each other. The relevance is referred as relations, correlations and interaction of data attributes [28]. The maximum and meaningful relevance between data attributes helps to classify the objects [29], finding batter prediction rules [37] and meaningful information and patterns [28][30]. Insignificant relevance increases the misclassification rates and also degrades the performance of prediction model.

We apply the necessary correction measures, if needed to improve the relevance between data attributes.

B. Data Division

The datasets used to build prediction model are called as model sets. The model set is further divide into three parts [25] [31]:

1. *Training Set,* used to build the prediction model.
2. *Validation Set,* used to validate the prediction model.
3. *Test Set,* used to determine the performance of prediction model on unseen data.

These three datasets contain all attributes of model set but strictly, they should not contain same values for all attributes [25] [31]. Therefore, all these three datasets contain different copy of model set and only used for defined purpose.

Why Training Set, Validation Set and Test Set are used separately?

As we already discussed that these three are different and cannot be used in place of each other. We can understand this by an example, if we ask about the strength of bridge to the same engineer who built it; he always replies that the strength of bridge is very good. That's why, to verify the truth, government makes a team. This team does not include those engineers who built the bridge. After verification, the bridge is not directly open for public transports. First, government tests the bridge strength by running those vehicles which have complete accidental precautions after then it open for all. Same in case of development of defect prediction model, the training set build the model so cannot validate and test the model. Validation set and test set are also different and used to validate the model and to check actual performance of prediction model respectively.

C. Model Building

To build the prediction model, first we select the appropriate prediction methodology and then build it using the training set.

1) Training Set: The training set is the input datasets in construction of defect prediction model. The model uses the training datasets to produce the interpretations for the existing defects in the targeted modules. These interpretations are accrued in the form of a neural network, a decision tree, a linkage graph, or some other representation depends on the prediction methodology [25].

2) Prediction Methodology: To build a prediction model, the appropriate selection of defect prediction methodology is very important task. There are various statistical and machine learning data mining methodologies have been developed for the purpose of defect prediction such as, C 4.5 algorithm, Bayesian technique, Logistic Regression and also, there is a lot more [5]. Thus, we select the suitable prediction technique and apply the training data and build the defect prediction model.

D. Model Finalization

Once the defect prediction model built, it is necessary to check the validity and also evaluate the performance of this model before applying it to solve defect prediction problems during software development.

1) Validate the Model: The validation confirms that the correct prediction model is developed [32]. It is used to determine that, how prediction model is accurate. Incomplete or inappropriate validation can accidentally mislead to both prediction results and expectations of software practitioners [33]. To performed validation on prediction model, first select an appropriate validation scheme and then apply it on defect prediction model by using the validation datasets.

a) Validation Scheme: Various studies [10] [11] [24] [31] [34] consider to n-fold cross validation as efficient approach, where the datasets are split into n folds. 10-fold cross validation is most common, where the datasets are split into 10 folds [9-11] [24] [28] [31].

b) Validation Set: The validation datasets must be different from the training set to obtain good performance in the optimization or selection stage [31]. The error rate on the validation set should be larger than the error rate on the training set [25].

2) Performance Evaluation of Model: After validating the prediction model, we check the prediction performance is satisfactory or not and also how this model will work on unseen data. The performance evaluation checks the ability, stability and success ratio of the prediction model [31]. The performance of model can be evaluated under certain measures and the test dataset is used as input datasets during this evaluation.

a) Performance Measures: The commonly used performance measures are accuracy, precision, recall and F-measure [9][24][11][35][31] but some studies consider them inefficient [12][36]. The other performance measures [13] for prediction model are Odds Ratio, Power, Probability Ratio, Gini Index, Mutual Information, Kolmogorov-Smirnov Statistic, Deviance, Geometric Mean, Area Under the ROC Curve, Area Under the Precision-Recall Curve and also many exist.

b) Test Set: The accuracy of performance evaluation of prediction model is depends upon, how they procured from model set. It would be much batter that the test set should come from different time period of software development then the training and validation set. In the ideal case, test set should come from more recent time period of software development; however, this is not often possible in practice [25]. Training set can never be used to evaluate the performance of prediction model [24] [25] [35] [31] [33].

c) Is Performance of Prediction Model Satisfactory?: If the consequent result of performance evaluation of prediction model is not satisfactory then re-evaluate the building process of prediction model and if any correction is needed, apply it. After making all corrections, again validate the model and also check its performance. In spite of all correction made, however the performance of prediction model is not improving then change the prediction methodology being used to build the model and assume that this prediction methodology is not feasible with the datasets because different prediction methodologies should chose for different datasets [33].

E. Apply the Model

If the performance of prediction model is satisfactory then the task of development of defect prediction model is completed. The model built after passing through various examinations, thus it should be good enough and thereby, the organization can use it to predict the defect during the software development.

V. EMPIRICAL RESULT ANALYSIS

As we already discuss that the defect prediction model build using publicly available defect data is not work efficiently when it used to predict the defects during software development of different organizations. To investigate our postulation we built the defect prediction model by using the J48 (based on C 4.5 data mining algorithm) Weka Tree Classifier Tool [40] [31] and 69 defect datasets one by one. These datasets are used the Chidamber and Kemerer (CK) object oriented metrics [38][39] to store the data values and publicly available on promisedata website [**7**]. So as per result the 69 models are developed.

We used all 69 models separately to predict the defects in AMT datasets and analysis the prediction result. Our analysis is based on Actual Prediction (AP) and False Alarm (FA), where AP is correct prediction, means the instance that are actually defective, the model predicted them defective (known as True Positive (TP)) and the instance that are not defective, the model predicted them non-defective (known as True Negative (TN)) and FA is incorrect prediction, means the instance that are actually not defective but the model predicted them defective (known as False Positive (FP)) and the instance that are actually defective but model predicted them non-defective (known as False Negative (FN)) . The AP is the sum of TP and TN and FA is the sum of FP and FN. The consequence of our analysis is:

1. The values of AP for 42 prediction models lie between 20 to 40% and value of FP lie between 60 to 80%.
2. The values of AP for 18 prediction models lie between 40 to 60% and value of FP lie between 40 to 60%.
3. The values of AP for 3 prediction models lie between 60 to 65% and value of FP lie between 35 to 40%.
4. The values of AP for 5 prediction models lie between 70 to 78% and value of FP lie between 22 to 30%.
5. The value of AP for only 1 prediction model is approximately 89.30% and value of FP approximately 10.70%.

In addition, we borrow the historical defect datasets from a java based Software Development Company and denominate these datasets as AMT because they are collected during the development of "Automated Messaging Tool" software product of this company. These datasets also used the CK object oriented metrics to store the data values and it contain 1047 instance where, 819 are defective and 228 are non-defective.

We divide the "AMT" datasets into two parts. First part of datasets contain 439 instance, where 352 are defective and 87 are not defective and second part contain 608 instance, where 467 are defective and 141 are not defective. We used first part of datasets to build the defect prediction model and apply it to predict the defects into second part of datasets. Again we used second part of datasets to build the model and apply it to predict the defects into first part of datasets. The same prediction methodology (J48 classifier of weka tool) is used to build the model. The consequence of this analysis is:

1. The value of AP for first prediction model is 91.28% and the value of FP 8.72%.
2. The value of AP for first prediction model is 90.43% and the value of FP 9.57%.

VI. THREAT FOR VALIDITY

We didn't follow the defined approach for building the defect prediction model completely in our analysis because our aim is not to build the efficient defect prediction model but the purpose is to show that, how a prediction model

work if it should be made by publicly available defect data. We also couldn't perform more analysis for defect prediction model built using organizational personal defect data due to lack of datasets.

VII. CONCLUSION

Our empirical analysis shows that the models built using publicly available defect data; only 8.7% models are usable whose values of AP are lie between 70 to 90%. The maximum value of AP for them is 89.30% whether the models built using organizational personal data work with outstanding performance and contain more than 90% value of AP.

In addition, Turhan et al. [14] stated that the organizations use the prediction models built using publicly available data, for predicting the defects in their own software development project, the probability of finding the defective modules may increases but abruptly the probability of FA rate is also increases. Zimmermann et al. [15] perform defect prediction on 622 different projects by using prediction models that are built using publicly available data, and found that only 3.4% actually worked.

So, the essence is that the models built using publicly available data, mostly not work well and if any organization used them, then unknowingly the development time and cost of the software increased due to high FA rates of these models.

Our study and empirical analysis motivates the software development organizations to set-up a data repository for collecting their own software development defect data and use these data to develop the defect prediction model, thereby they can improve the defect prediction approach during their software development. The improvement in defect prediction approach would be result as less development time and cost, and high software product quality. Our generalized approach facilitates to software development organization who want to develop the effective defect prediction model.

ACKNOWLEDGMENT

I pay homage to my mother Late Smt. Vimla Singh. She vitalizes me. She used to care me, love me, and give me such strength whereby I can fight with all hardness of life. I cannot complete my any work without memorize her. I also cannot explain her contributions for me and also cannot explain how much importance she has for me.

REFERENCES

[1] Deming, W. E., Out of the crisis: quality, productivity and competitive position, Cambridge University Press, 1988.

[2] Bessin, G., The business value of software quality, IBM Rational, 2004.

[3] Humphrey, W. S., Learning from Hardware: Design and Quality, Software Engineering Institute.

[4] Gilb, T., and Graham, D., Software Inspection, Addison-Wesley, 1993.

[5] Singh, A., and Singh, R., "Assuring Software Quality using Data Mining Methodology: A Literature Study", Proceedings in International Conference on Information Systems & Computer Networks (ISCON), 2013, pp. 108–113.

[6] http://www.bugzilla.org/

[7] http://promisedata.googlecode.com/svn/trunk/defect/

[8] http://promise.site.uottawa.ca/SERepository/datasets-page.html

[9] Koru, G., and Liu, H., "Building Effective Defect-Prediction Models in Practice", Published in IEEE Software, IEEE Computer Society Press Los Alamitos, CA, USA, 2005, vol. 22, pp. 23-29.

[10] Rodriguez, D., Ruiz, R., Cuadrado-Gallego, J., and Aguilar-Ruiz, J., "Detecting fault modules applying feature selection to classifiers," Proceedings in IEEE International Conference on Information Reuse and Integration, 2007, pp. 551–557.

[11] Hall, T., Beecham, S., Bowes, D., Gray, D., and Counsell, S., "A Systematic Literature Review on Fault Prediction Performance in Software Engineering", Published in IEEE Transactions on Software Engineering, 2012, vol. 38, pp. 1276–1304.

[12] Menzies, T., Greenwald, J., and Frank, A., "Data mining static code attributes to learn defect predictors", IEEE Transactions on Software Engineering, 2007, pp. 637-640.

[13] Khoshgoftaar, T. M., Gao, K., and Napolitano, A., "A Comparative Study of Different Strategies for Predicting Software Quality", Proceedings in the 23rd International Conference on Software Engineering & Knowledge Engineering (SEKE'2011), Eden Roc Renaissance, Miami Beach, USA, 2011.

[14] Turhan, B., Menzies, T., Bener, A. B., and, Stefano, J. D., "On the relative value of cross-company and within-company data for defect prediction", Published in Empirical Software Engineering, , 2009, vol. 14, pp. 540–578.

[15] Zimmermann, T., Nagappan, N., Gall, H., Giger, E., and Murphy, B., "Cross-project Defect Prediction: A Large Scale Experiment on Data vs. Domain vs. Process", Proceedings in the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on the foundations of software engineering, ACM New York, NY, USA, 2009, pp. 91-100.

[16] Brooks, F. P., The Mythical Man-Month, Anniversary (2nd Edition), Addison-Wesley Publication, 2002.

[17] Shewhart, W. A., "Economic control of quality of manufactured product", D. Van Nostrand Company, Inc., 1931.

[18] Basili, V. R., and Perricone, "B. T., Software Errors and Complexity: An Empirical Investigation", Communications of the ACM 27, 1984, pp. 42-52.

[19] Basili, V.R., and Selby, R.W., "Comparing the Effectiveness of Software Testing Strategies", Published in IEEE Transaction on Software Engineering, vol. 13, 1987, pp. 1278-1296.

[20] Wixom, B., and Watson, H. J., "An empirical investigation of the factor affecting data warehouse success", Published in MIS Quarterly, vol. 25, 2001, pp. 17-41.

[21] Schmidt, R., Lyytinen, K., Keil, M., and Cule, P., "Identifying the software project risks: an international Delphi study", Published in Journal of Management Information Systems, 2001, pp. 5-36.

[22] Jiang, J. J., Klein, G., and Balloun, J. L., "Ranking of system implementation success factors", Published in Project Management Journal, 1996, pp. 50-55.

[23] Kim, C. S., and Peterson, D. K., "A comparison of the perceived importance of information systems development strategies by developers from the United States and Korea", Published in Information Resource Management Journal, vol. 16, 2003, pp. 1-18.

[24] Han, J., and Kamber, M., Data Mining: Concepts and Techniques, Morgan Kaufmann, 2000.

[25] Berry, M. J. A., and Linoff, G. S., Data Mining Techniques: For Marketing, Sales, and Customer Relationship Management, Wiley Publishing, Inc., Indianapolis, Indiana, 2004.

[26] Wang, H., Khoshgoftaar, T. M., Gao, K. and Seliya, N., "Mining data from multiple software development projects", Proceedings in IEEE International Conference on Data Mining Workshop, ICDMW '09, 2009, pp. 579–606.

[27] Gao, K., Khoshgoftaar, T. M., Wang, H., and Seliya, N., "Choosing software metrics for defect prediction: an investigation on feature selection techniques", Published in Journal Software: Practice and Experience, 2011, pp. 935–942.

[28] Peng, H., Long, F., and Ding, C., "Feature Selection Based on Mutual Information: Criteria of Max-Dependency, Max-Relevance, and Min-Redundancy", Published in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 27, 2005, pp. 1226 – 1238.

[29] Farhadi, A., Endres, I., Hoiem, D., and Forsyth, D., "Describing Objects by their Attributes", IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2009, pp- 1778 – 1785.

[30] Ke, Y., Cheng, J., and Ng, W., "Correlated Pattern Mining in Quantitative Databases", Published in ACM Transactions on Database Systems (TODS), vol. 33, 2008.

[31] Witten, I. H., and Frank, E., Data Mining: Practical Machine Learning Tools and Techniques, Morgan Kaufmann Publishers, Elsevier, 2005.

[32] Runeson, P., Andersson, C., Thelin, T., Andrews, A., and Berling, T., "What Do We Know about Defect Detection Methods", Published in IEEE Software, vol. 23, 2006, pp. 82–90.

[33] Song, Q., Jia, Z., Shepperd, M., Ying, S., and Liu, J., "A General Software Defect-Proneness Prediction Framework", Published in IEEE Transactions on Software Engineering, vol. 37, 2011, pp. 356–370.

[34] Briand, L. C., Emam, K. E., Freimut, B. G., and Laitenberger, O., "A Comprehensive Evaluation of Capture-Recapture Models for Estimating Software Defect Content", Published in IEEE Transactions on Software Engineering, vol. 26, 2000, pp. 518–540.

[35] Arisholm, E., Briand, L. C., and Johannessen, E. B., "A systematic and comprehensive investigation of methods to build and evaluate fault prediction models", Published in The Journal of Systems and Software, Elsevier Science Inc. New York, NY, USA, vol. 83, 2010, pp. 2-17.

[36] Menzies, T., Dekhtyar, A., Distefano, J., and Greenwald, J., "Problems with precision", Published in IEEE Transactions on Software Engineering, 2007, pp. 2 – 13.

[37] Lee, Y. K., Kim, W. Y., Cai, Y. D., and Han, J., "CoMine: Efficient Mining of Correlated Patterns", Proceedings in The 3rd IEEE International Conference on Data Mining (ICDM), 2003, pp. 581–584.

[38] Chidamber, S. R., and Kemerer. C. F., "A metric suite for object oriented design", Published in IEEE Transactions on Software Engineering, 1994, vol. 20, pp. 476 – 493.

[39] Chidamber, S. R., Darcy D. P., and Kemerer. C. F., "Managerial use of metrics for object-oriented software: an exploratory analysis" Published in IEEE Transactions on Software Engineering, 1998, vol. 24, pp. 629-639.

[40] http://www.cs.waikato.ac.nz/ml/weka/