READING ASSIGNMENT III

A COMPARATIVE ANALYSIS OF THE EFFICIENY OF CHANGE METRICS AND STATIC CODE ATTRIBUTES FOR DEFECT PREDICTION

By - Valentin Dallmeier and Thomas Zimmermann

II. IMPORTANT KEYWORDS

ii1. Cost Sensitive Classification

Classification problem such as fraud detection, medical diagnosis are cost sensitive meaning cost of missing a target is much higher than cost of reporting a wrong data.

ii2.Hit Rate Caching

List files that have most recently modified file change.

ii3.Software Metrics

Standard of measure of a degree to which a software system or process possesses some property.

ii4.Confusion Matrix

It is *used to describe the performance of a classification model on a set of test data for which the true values are known. It has variables true positive, false negative, true negative and false neagtive.*

III. BRIEF NOTES

iii1. Motivation

A comparative analysis like this on different software models can help us decide the metrics that are easy to collect during early phase of software development, which model, predictor should be used for which product and how accurate is that model for a software system.

iii2. Related Work

Weyukar et al.[1] have focused on impact of software process on defectiveness of a software. Menzies et al.[2] have worked to find out the relationship between software defects and code metrics. Graves et. al [3] have argued that process metrics are more effective to predict defectiveness rather than code metrics in their work. However, authors feel none of these could be considered conclusive.

iii3.Commentary

The authors have used the below for their study:-

- *data for eclipse software release available in PROMISE repository(www.promisedata.org/repository). The authors have chosen to use 18 metrics from available 198 metrics.*
- *they have used "revision history" or "refactor" like comments in software push to find the code changes.*
- *They have used formula to calculate the "weighted age" which is summation of age Max change set and average changeset.*
- *They have used that data to create confusion metrics and then used for their models.*

iii4.Results

The authors have suggested below guidelines based on their study:-

- *the pre or post release defect distribution might give the first clue of where the defects are.*
- *using data from revision management systems, we can build a simple prediction model first.*
- *if this model does not give proper result, we can use the metrics like weighted age, etc used in study to build our model.*
- *if recall is low, we can tune the model using cost sensitive classification and determine cost matrix.*
- *if all the other fail, use combined metrics instead of code model.*

IV. IMPROVEMENTS

iv1. The file search history results can be considered as a parameter for this study.

Relation to Original Paper

The original paper uses this paper as the reference to provide evidence of comparative analysis of the studied models.

References:-

1. *Using Developer Information as a Factor for Fault Prediction - Weyukar JE, Ostrand TJ, Bell RM - 2007*
2. *Data Mining Static Code Attributes to Learn Defect Predictors - Menzies T, Greenwald J, Frank A - 2007*
3. *Predicting Fault incidence using software change history - Graves TL, Karr AF, Marron JS - 2000*