Journal of Web Engineering, Vol. 16, No.3&4 (2017) 293-310
© Rinton Press

PREDICTION OF DEFECT DENSITY FOR OPEN SOURCE SOFTWARE USING REPOSITORY METRICS

DINESH VERMA

Jaypee University of Engineering and Technology, Guna, India
dinesh.hpp@gmail.com

SHISHIR KUMAR

Jaypee University of Engineering and Technology, Guna, India
dr.shishir@yahoo.com

Received September 9, 2015
Revised December 26, 2016

Open source software refers to software with unrestricted access for use or modification. Many software development organizations are using this open source methodology in their development process. Many software developers can work in parallel with the open source project using the web as a shared resource. The defect density of such projects is often required to be predicted for the purpose to ensure quality standards. Static metrics for defect density prediction require extraction of abstract information from the code. Repository metrics, on the other hand, are easy to extract from the repository data sets. In this paper, an analysis has been performed over repository metrics of open source software. Further, defect density is being predicted using these metrics individually and jointly. Sixty two open source software are considered for analysis using Simple and Multiple Linear Regression methods as statistical procedures. The results reveal a statistically significant level of acceptance for prediction of defect density using few repository metrics individually and jointly.

Key words: Defect Density, Repository Metrics, Simple and Multiple Linear Regressions.
Communicated by: M. Gaedke & Y. Deshpande

1 Introduction

Open source software becoming important for software industries. These types of software system are being used by both business professionals and academician in their work. Such projects are mostly developed outside the companies, by the volunteers. The development methodologies of such type of projects are quite different from the methodology used for commercial projects. The developers of these projects can work in parallel using the different shared resources during development, consequently, more individuals and companies are involved in the development process for open source projects. Unlike traditional commercial development methodologies, deadlines and number of assignments are not fixed in the development of open source software. Using open source software development methodology, high level of quality can be achieved in comparison to traditional commercial development methodology [1, 22]. Defect density is the parameter that can be used to assess sensitivity of defect for specific software. The predication of defect density for open source software helps in maintaining an acceptable level of quality for further versions of software. Most of the researches for defect density prediction of open source software are based on the static metrics of projects. Chidamber and Kemerer defined object oriented static metrics of the software [3]. The analysis of such type of metrics required lots of information from the source code files of the project. Extraction of information for these static metrics from the abstracted data requires lots of efforts. An approach for prediction of defect density of open source software with analysis of information that can be easily extracted from the project is need of time. Repository metrics like, size of project, number of bugs, total number of downloads, number of developers, etc. are easily available on the project pages. This information can be easily made available for analyzing the approach for defect density prediction. In this paper relationship between different repository metrics of open source software with defect density has been established. This relationship can be used by the developer of open source software to predict the defect density of next version and maintain the acceptable level of quality. In this paper five repository metrics have been identified to predict the defect density of open source software. Initially, the prediction of defect density by individual metrics, and then prediction with all metrics jointly has been discussed. The proposed approach has been analyzed with 62 open source software projects available at SourceForge.net [2]. The simple linear regression method has been used to predict defect density by repository metrics independently. The multiple linear regression method has been used to predict the defect density by repository metrics jointly. Both simple and multiple regression results show a statistically significant prediction of defect density by repository metrics independently and jointly.

This paper is organized as follows. In section 2, the related work carried out in terms to defect density prediction by the different factors of open source software is discussed. In section 3, repository metrics of open source software used in this research are discussed. In section 4, formulation of research hypotheses, data collection, and research methodology are discussed. In section 5, experimentation on the collected data for proving the research hypotheses is discussed. In section 6, the summary of results comes from different experimentation have been discussed and compared with the previous works discussed in section 2. In section 7, possible threats to validity for this research are discussed. Conclusions and future works directions are mentioned in section 8.

2 Related Work

Static metrics have been used in most of the research with open source software for prediction of defect density. In the study of Verma and Kumar [19], the module size of software has been used to predict the defect density. In another study Verma and Kumar [20] shows exponential relationship of defect density with module size where the power of size is not restricted to integer values.

In another research, static code metrics have been used to predict the defect density using available public data sets [24]. Very limited work has been carried out in the establishment of a relationship between defect density and repository metrics of open source software. Study by Weyuker, Ostrand, and Bell [4] concludes that many developers do not have a significant impact on the prediction of defect density. They rely on their analysis of four large software projects from the industries. In this approach negative binomial regression method has been used for prediction of faults. A significance impact on software quality by risk assessment has been modeled based on Model-View-Controller (MCV) software architecture [25].

Subramanyam and Krishnan [7] mentioned significant association of object oriented metrics given by Chidamber and Kemerer [3] object oriented metrics suite with defects. They have concluded that even after controlling the size of software, these metrics show the implications on software defects.

In a study, suitable method to calculate the object oriented metrics of open source software have been described [5]. Using logical and linear regression statistical methods they assess the applicability of the different object oriented metrics to predict the number of bugs in classes. Their analysis was based on the seven released versions of Mozilla's. A semi-supervised learning ROCUS has been proposed that address the two major practical issues in software defect detection. First, it is rather difficult to collect a large amount of labeled training data for learning a well-performing model; second, in a software system there are usually much fewer defective modules than defect-free modules, so learning would have to be conducted over an imbalanced data set [23].

Knab, Pinzger, and Bernstien [6] used source code, modification, and defect measure to predict the defect density of Mozilla's open source projects. They have applied the Data mining approaches on these metrics for analysis of their hypothesis. The study shows a small prediction of defect density based on lines of code.

Mockus, Fielding, and Herbsleb [8] show the importance of developer participation in the development of two large open source software projects the Apache web server and Mozilla browser. Five user's perspective naming Users' expectations, Usability bug reporting, Interactive Help Features, Usability learning and, usability guidelines have great influence on the open source software usability [21].

Sherriff, Williams, and Vouk [9] proposed a method for estimating defect density of Haskell programs by using in-process metrics suite. The multiple linear regression statistical analysis shows a significant prediction of defect density by static metrics.

Rahmani and Khazanchi [10]used the repository metrics of open source software to predict the defect density. They have used only three repository metrics as a predictor of defect density. The number of commit of projects has not been considered in their study.

In another study of open source projects Caglayan, Bener, and Koch [11] focused on improving the prediction performance of achieving lower probability of false alarm rate of defect predictor. They also discuss the importance of repository metrics over the other metrics of software.

In all the research work mentioned above on open source software, prediction of defect density does not indicate the relationship of defect density with other repository metrics. In proposed work, the relationship has been established of defect density with some important repository metrics of open source software.

3 Repository Metrics

Static metrics have been used in most of the research with open source software for prediction of defect density. In the study of Verma and Kumar [19], the module size of software has been used to predict the defect density. In another study Verma and Kumar [20] shows exponential relationship of defect density with module size where the power of size is not restricted to integer values.

In most of the previous research, traditional static code attributes are used for modelling software data for their research analysis. Although, static code attributes can automatically be extracted from the source code, the information is abstracted in the database in terms to maintain the authenticity of information. Extraction of static code attributes requires more effort; there is a need to explore alternative metric sets such as repository metrics that can be easily extracted from the project web pages. In the proposed approach, five metrics have been identified, and analysis has been performed to establish the relationship between these metrics and defect density of open source software. The conceptual definition of each metric is as follows:

a. *Software Size:* - Size measures of software have direct application to the planning, tracking, and estimating the software projects [12]. Software size can be measured in Lines of Code (LOC) or can be measured in number of Function Points (FPs). In the work presented through this paper Lines of Code has been used to measure the size of software. The size has been considered for analysis that is available on the project page at the time of data collection.

b. *Defects:* - According to the IEEE standard classification of software anomalies [13], defects are imperfection or deficiency in a work product that work product does not meet its requirement or specifications and needs to be either repaired or replaced. The total number of software defects has been taken from the project statistics page for analysis of this study. Total numbers of defects are the numbers that have been collected at the time of data collection.

c. *Number of Developers: - Developers* of the open source software are defined as the total number of persons those download the project and contribute some modification/updating in the project. Individual developers make modification/updating in the project according to their own experience and expectation of the project. These types of modifications vary when the developers varies. The total number of developers of project for analysis has been taken from the project page at the time of data collection.

d. *Number of Downloads:* - This is the total number of downloads for the particular projects given in the project statistics at the time of data collection.

e. *Commits:* - In open source software commits basically record the changes made by developers in the source code of the project [14]. These changes take place when the developer makes any addition, deletion, and/or modification in the source code.

f. *Defect Density: - Defect* density of software can be defined as the ratio of total number of defects with total size of software. In this study the size has been taken as a KLOC for calculating defect density.

4 Research Hypotheses

As mentioned in previous sections, the work presented through this paper focused to establish the relationship between repository metrics of open source software with the defect density. The relationship further used to predict the value of defect density with these predictors individually and jointly. This prediction can be used to control the quality of open source software at an acceptable level. In the proposed approach five repository metrics have been used such that software size, total defects, number of developers, number of downloads, and number of commits. To accomplish the objective, six hypotheses have been designed to address the relationship of repository metrics with defect density in concern with open source software.

4.1. Formulation of Research Hypotheses

Following six hypotheses have been formulated to analyze the impact of five repository metrics on the defect density:

Hypothesis 1 (H1): - Software size of open source software has a negative relationship with Defect Density.

In open source software the size measured as a Kilo line of code (KLOC). Most of study shows that when the size of traditional software increases, defect density will also increase. In a research by Basili and Perricone [15] shows that defect density will decrease when the size is increasing at a certain level. The analysis proposed by Basili's study deals with the small size of modules only. So a relationship needs to be developed which can cater to the need for all possible sizes of models and used for prediction of defect density for further similar projects. In the proposed work a relationship of the size of project has been established with defect density.

Hypothesis 2 (H2): - Number of defects has a positive relationship with Defect Density in Open Source Software.

Open source software is openly used and modify by the many numbers of users/developers. Although, the number of defects is increasing as the modifications are made to the project's, defects are found and fixed very quickly in open source software because there are more user / developers looking for the problem. These activities may lead to increase the size of updated version of open source project in terms to solve the problem or fixing the defects. So a relationship required to be established between the total numbers of defects with the defect density.

Hypothesis 3 (H3): - More Number of developers in an Open source project has lower defect density.

Number of developers increases with time in open source projects. These developers focus to identify the problem and make some changes to repair or fix the problem in the projects. Individual developers make modification/updating in the projects according to their own experience and expectation of the project. These types of modifications vary when the developers varies. To evaluate the impact of different developers on defect density, a relationship of defect density required to be established with numbers of developers in open source projects.

Hypothesis 4 (H4): - More downloads of open source software increase the defect density.

Number of downloads for a particular open source software, indicate the usability and effectiveness of projects among the software users. A relationship of defect density required to be established with the number of downloads.

Hypothesis 5 (H5): - Number of commits has a positive relationship with defect density in open source projects.

During the software evaluation, continues changes are made to the systems those are stored in the version control systems of the software. These changes are termed as the commits made by different developers. These changes may lead to insert new defects in the software. So these commits can directly affect the defect density of the software. A relationship required to be established between defect density and number of commits make by different developers.

Hypothesis 6 (H6): - Software size, number of defects, number of developers, number of downloads and number of commit are jointly related to defect density.

The combined effect of software size, number of defects, number of developers, number of downloads, and number of commits on the defect density needs to be analyzed. Composite predictions of defect density by all the metrics jointly, need to be discussed. So the model has been formulated that uses all the five repository metrics as predictors for the defect density.

4.2. Data Collection

In this study six hypotheses have been formulated in section 4.1, these hypotheses required to be analyzed on the real data from the open source projects. For analysis of proposed hypotheses 62 open source projects have been randomly collected from the SourceFourge.net ("SorceForge", 2015), with the following considerations:

a) Projects having the 80% and above recommendation have been selected.

This recommendation means that the project has been used by 100 persons, and 80 persons have rated this project as a good project and recommend it.

b) Projects developed using programming languages as JAVA.

There are so many projects available in different languages. In this study, we have filtered out only JAVA based projects.

c) Software size has been calculated for Windows environment.

The projects developed on the windows environment have been selected. Size of projects is not dependent on operating environments.

d) Bugs data are clearly available on the project statistics page.

The project those satisfying above three conditions, but the details of bugs data are not clearly available, have been rejected.

The data collected from the project web page as an overall project not for a single release, such as Size of project, total number of defects, total number of commits, total number of downloads, and total number of downloads. All the metrics values taken from the project web pages are available on the date of data collection.

The summary of descriptive statistics for Sixty two projects is given in Table 1.

Table 1: Descriptive statistics of dataset of 62 projects

Metric	**Mean**	**Median**	**Standard deviation**	**Maximum**	**Minimum**
Software Size	178.56	84.2	262.78	1391	3
Number of Defects	873.36	408.5	1123.8	4912	18
Number of Developers	14.98	10	14.01	61	1
Number of Downloads	2238.86	1563.5	2165.9	9389	277
Number of Commits	3063.36	1136.5	4315.36	18121	9
Defect Density	7.28	4.19	7.15	42.89	1.06

4.3. Research Methodology

To analyze above six hypotheses, linear regression statistical method has been used to prove these hypotheses. For linear regression method SPSS tool [16] has been used. In the above mentioned relationship, simple linear regression method has been used for establishing the relationship of each individual predictor. Further, multiple linear regression method has been used for showing the joint impact of predictors on defect density.

Before performing the linear regression with the data set, first normality test has to be performed [17,18]. The information randomly collected from the SourceFourge, could not clear the normality test.

So initially the logarithm value of each metric has been taken and checked for the normality. All the metrics data pass the normality test with their logarithmic value.

After performing the normality test simple linear regression and multiple linear regressions have been performed for individual and joint effect respectively.

5 Experimental Result

As discussed in the previous section, first the normality test has to be performed with the logarithmic value of each metrics data set. The normality tests are supplementary to the graphical assessment of normality. The main tests for normality are Kolmogorov-Smirnov (K-S) test and Shapiro-Wilk (S-W) test (Elliot & Woodward, 2007). The K-S test is performed if the sample size is more than 2000, and the S-W test is performed for the sample size less than 2000 (Elliot & Woodward, 2007). The sample size in this research is only 62, so only S-W values for normality are considered.

The result of normality tests for all considered metrics are summarized in Table-2 and corresponding Graph for Normality Test of all metrics shown in Figure 1.

Table 2: Normality Test Results of all metrics

Metric	**Kolmogorov-Smirnov**			**Shapiro-Wilk**		
	Statistic	**df**	**Sig.**	**Statistic**	**df**	**Sig.**
Size of Project	.072	61	.200	.985	61	.669
No. of Defects	.089	61	.200	.974	61	.216
No. of Developers	.055	61	.200	.976	61	.265
No. of Downloads	.106	61	.080	.965	61	.077
No. of Commits	.085	61	.200	.961	61	.048

The Table-2 indicates the statistical result of normality test. In this result Sig. Value (also call this as p value) is greater than 0.05 (Elliot & Woodward, 2007), it indicates that the concern metrics values are normally distributed. Figure 1 shows the graphs of normal distribution of all metrics values. In each graph most of the points are conceding on the line that indicates the normal distribution.

According to the result of normality tests of individual metrics data set, the p value for all metrics data set is equal or greater than 0.05. Now linear regression statistical test can be performed on the logarithmic value of each metrics data set.

The regression tool in SPSS provides the result for linear regression test in terms of different parameters. Regression analysis is used to describe the response of a dependent variable which changes according to magnitude of the independent variables. It is one of the form of inferential statistical analysis which indicates the relationship between one dependent variable and one or more independent variables.

R^2 is a statistical measure of how close the data are to the fitted regression line, 100% indicates that the model explains all the variability of the response data around its mean. Like R^2 adjusted R^2 also indicates how well terms fit a curve or line, but unlike it adjusts for the number of terms in a model. Adding more number of useless variables to a model, adjusted R^2 will decrease and adding more useful variables will result an increase in adjusted R^2 value. The Standard Error of the Estimate is the standard deviation of the data about the regression line. It is a measure of the variability of predictions in regression.

Analysis of Variance (ANOVA) provides information about levels of variability within a regression model and forms a basis for tests of significance. Sum of Squares represents the measure of variation from the mean. The degree of freedom (df) is the number of variables that are free to vary. It is one less than the number of parameters being estimated. Mean squares are estimates of variance across groups. The F ratio is the ratio of two mean square values.

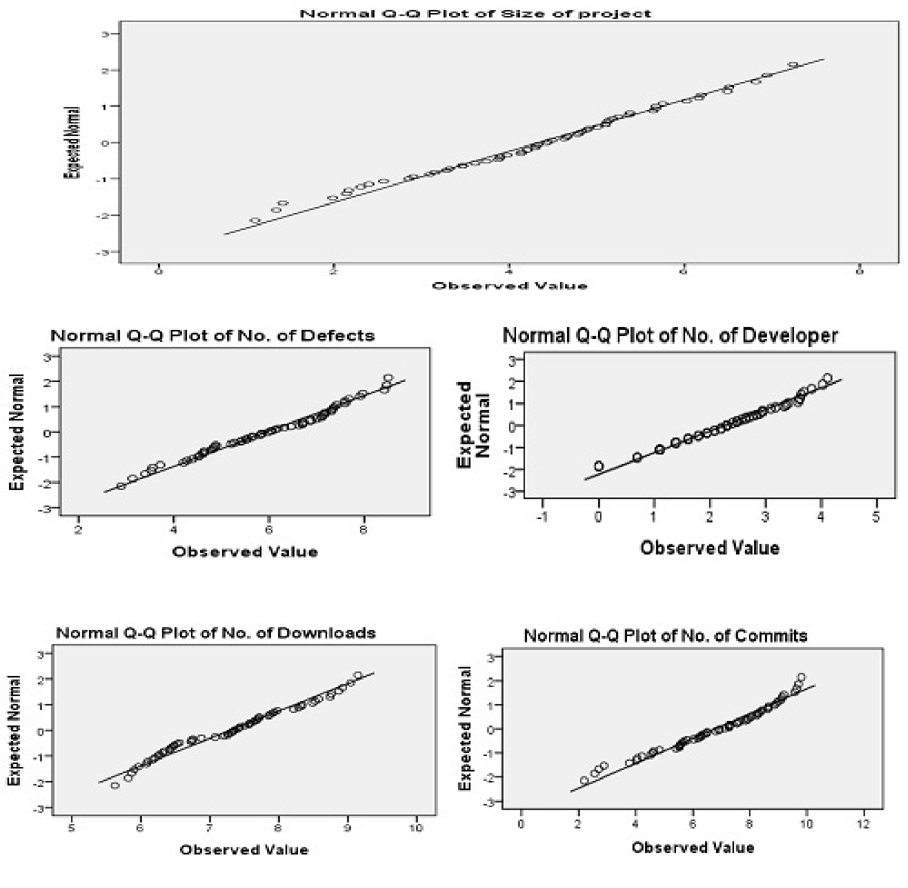

Figure 1: Normality Graph for all Metrics

Unstandardized regression coefficients (B) show the average change in the dependent variable associated with a 1 unit change in the dependent variable, other variables being constant. However, Standardized beta coefficients (Beta) are the estimations that have been standardized so that the variances of dependent and independent variables are 1. T statistic is a ratio of the departure of an estimated factor from its theoretical value and its standard error. It represents the change between the mean scores of two groups, while considering any variation in scores. R-squared value, significance of the observed regression line (p- value). The p- value is the significance level that is used to accept or reject the hypothesis. P-value defined as a probability of error that involved in accepting our observed result as a valid result.

To accept the hypothesis, the p-value must be less than or equal to 0.05, otherwise reject the hypothesis (Elliot & Woodward, 2007). The R-squared value represents the percentage extent of variation in the dependent variable presented by the independent variable (Elliot & Woodward, 2007). For example, in a linear regression result, the *p*-value 0.03 indicates that, accepting variability of dependent variable by independent variable, only 3% of the population does not valid. The R-squared value 0.16 indicates that 16% of variability presented by the independent variable.

Now each hypothesis have been analyzed using the linear regression statistical method and following results have been generated:

5.1 Analysis for H1

After performing the regression on the SPSS tool with taking Defect Density as dependent variable and Size of Project as predictor, the results are shown in Table 3.

Table 3: Simple Linear Regression result for H1

Model Summary				
Model	**R**	**R Square**	**Adjusted R Square**	**Std. Error of the Estimate**
1	0.294	0.86	0.071	0.82990

ANOVA						
Model		**Sum of Squares**	**df**	**Mean Square**	**F**	**Sig.**
1	Regression	3.896	1	3.896	5.656	0.021
	Residual	41.324	60	0.689		
	Total	45.220	61			

Coefficients						
Model		**Unstandardized Coefficients**		**Standardized Coefficients**	**t**	**Sig.**
		B	**Std. Error**	**Beta**		
1	(Constant)	2.383	0.342		6.974	0.000
	Size of project	-0.178	0.075	-0.294	-2.378	0.021

As shown in the Table-3, the R-Squared value indicates that 8.6% of variability of defect density predicted by the Size of Project. It also indicates that some other factors predict the remaining 91.4% of variability. The *p*- value is 0.021 that is less than 0.05. So this hypothesis can be accepted. The negative value of B for Size of the project indicated that defect density has the negative relation with Size of the project. So when the sizes will increases the defect density will be decreases.

5.2 Analysis for H2

Now performing the regression on the SPSS tool with taking Defect Density as dependent variable and No. of Defects as predictor, the results are shown in Table 4.

As shown in Table-4, the R-Squared value indicates that 9.7% of variability of defect density predicted by the No. of Defects. It also indicates that some other factors predict the remaining 90.3% of variability. The *p*- value is 0.014 that is less than 0.05. So this hypothesis can be accepted. The negative value of B for Number of Defects indicated that defect density has the negative relation to Number of defects.

5.3 Analysis for H3

Now performing the regression on the SPSS tool with taking Defect Density as dependent variable and No. of Developers as predictor, the results are shown in Table 5.

As shown in Table-5, the R-Squared value indicates that 7.4% of variability of defect density predicted by the Number of developers. The *p*- value is 0.032 that is less than 0.05 indicates this hypothesis can be accepted. The negative value of B for Number of developers the project indicated that defect density is negatively related with this metric.

5.4 Analysis for H4

For analyzing H4, No. of Downloads has been used as a predictor on SPSS tool. The dependent variable will remain Defect Density. The results of this analysis are shown in Table-6.

According to the Table-6, the p-value is 0.353 that is greater than 0.05, hence the hypothesis has been rejected. The significance of this relationship is very less that indicates the inability in defect density prediction by number of downloads.

5.5 Analysis for H5

For analyzing H5, No. of Commits has been used as a predictor on SPSS tool. The dependent variable will remain Defect Density. The results of this analysis are shown in Table-7.

According to results from Table-7, the p-value is greater than 0.05, hence the hypothesis has been rejected. This is similar to the hypothesis H4, indicating the inability of Number of commits to predict the defect density.

Table 4: Simple Linear Regression result for H2

Model Summary

Model	**R**	**R Square**	**Adjusted R Square**	**Std. Error of the Estimate**
1	0.312	0.097	0.082	0.82475

ANOVA

Model		**Sum of Squares**	**df**	**Mean Square**	**F**	**Sig.**
1	Regression	4.407	1	4.407	6.479	0.014
	Residual	40.813	60	0.680		
	Total	45.220	61			

Coefficients

Model		**Unstandardized Coefficients**		**Standardized Coefficients**	**t**	**Sig.**
		B	**Std. Error**	**Beta**		
1	(Constant)	0.489	0.453		1.081	0.284
	No. of Defects	0.189	0.074	0.312	2.545	0.014

Table 5: Simple Linear Regression result for H3

Model Summary

Model	**R**	**R Square**	**Adjusted R Square**	**Std. Error of the Estimate**
1	0.272	0.074	0.059	0.83530

ANOVA

Model		**Sum of Squares**	**df**	**Mean Square**	**F**	**Sig.**
1	Regression	3.365	1	3.365	4.810	0.032
	Residual	41.864	60	0.698		
	Total	45.220	61			

Coefficients

Model		**Unstandardized Coefficients**		**Standardized Coefficients**	**t**	**Sig.**
		B	**Std. Error**	**Beta**		
1	(Constant)	2.134	0.261		8.168	0.000
	No. of Developer	-0.231	0.106	-0.272	-2.193	0.032

Table 6: Simple Linear Regression result for H4

Model Summary				
Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	0.120	0.014	-0.002	0.86187

ANOVA						
Model		Sum of Squares	df	Mean Square	F	Sig.
1	Regression	0.651	1	0.651	0.876	0.353
	Residual	44.569	60	0.743		
	Total	45.220	61			

Coefficients						
Model		Unstandardized Coefficients		Standardized Coefficients	t	Sig.
		B	Std. Error	Beta		
1	(Constant)	0.805	0.868		0.927	0.358
	No. of Downloads	0.110	0.118	0.120	0.936	0.353

Table 7: Simple Linear Regression result for H5

Model Summary				
Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	0.143	0.021	0.004	0.85918

ANOVA						
Model		Sum of Squares	df	Mean Square	F	Sig.
1	Regression	0.928	1	0.928	1.257	0.267
	Residual	44.292	60	0.738		
	Total	45.220	61			

Coefficients						
Model		Unstandardized Coefficients		Standardized Coefficients	t	Sig.
		B	Std. Error	Beta		
1	(Constant)	2.042	0.400		5.106	0.000
	No. of commits	-0.064	0.057	-0.143	-1.121	0.267

5.6 Analysis for H6

In the previous linear regression analysis, only one predictor has been considered for predicting the defect density. In the hypothesis 6, joint impacts of all five predictors on the defect density need to be analyze. For this analysis, multiple linear regression has been performed on the SPSS by selecting all five predictor together for finding the prediction rate of defect density. The multiple linear regression results have been shown in the Table 8.

From the Table-8 it is indicated that all the metrics jointly predict the 38.3% variability of defect density. This multiple regression gives a model in which all the predictor shows the relationship with defect density. This model is depicted by following Equation

Defect Density = 6.126 + 0.001(No. of Downloads) – 0.005(Size of project) + 0.004(No. of Defect) - 0.210 (No. of Developers) + 0.000(No. of Commits) ……. (1)

From the equation (1), it can be inferred that all the metrics predict the variability of defect density jointly, but the No. of Commits plays a very small role to predict the variation in defect density. The multiplicative factor of the Number of commits is very small that is 0.000004.

Table 8: Multiple Linear Regression result for H6

Model Summary

Model	**R**	**R Square**	**Adjusted R Square**	**Std. Error of the Estimate**
1	0.619	0.383	0.328	5.86264

ANOVA

Model		**Sum of Squares**	**df**	**Mean Square**	**F**	**Sig.**
1	Regression	11195.567	5	239.113	6.975	0.000
	Residual	1924.752	56	34.371		
	Total	3120.319	61			

Coefficients

Model		**Unstandardized Coefficients**		**Standardized Coefficients**	**t**	**Sig.**
		B	**Std. Error**	**Beta**		
1	(Constant)	6.126	1.482		4.134	0.000
	No. of Downloads	0.001	0.000	0.192	1.763	0.083
	Size of project	-0.005	0.007	-0.194	-0.794	0.431
	No. of Defects	0.004	0.001	0.614	4.865	0.000
	No. of Developer	-0.210	0.124	-0.412	-1.698	0.095
	No. of Commits	0.000	0.000	0.082	0.512	0.611

According to the Table-8, R-squared value indicates that 38.3% variability of defect density predicted by the all the five metrics jointly, and the relationship of all these five metrics with defect density is indicated in equation (1). This equation can be treated as a model for predicting defect density by all five metrics discussed in this research.

6 Summary of Results and Performance Evaluation

In this section, on the basis of analysis performed in the previous sections, summary of acceptance for each hypothesis has been tabulated in the Table 9.

According to the result for H1, the hypothesis cannot be rejected because of p-value that is nearly acceptable value. The analysis of H1 indicates that the Software Size is directly related to the Defect density. The percentage of defect density prediction by Size is very low because in the smaller size of software, it shows the declining trend with defect density [15].

Table 9: Summary of Hypotheses Results

Hypothesis ID	**R-squared Value**	**p-value**	**Result**
H1	0.086	0.021	Not Rejected
H2	0.097	0.014	Not Rejected
H3	0.074	0.032	Not Rejected
H4	0.014	0.353	Rejected
H5	0.021	0.267	Rejected
H6	0.383	0.000	Not Rejected

According to the result for *H1*, the hypothesis cannot be rejected because of *p*-value that is nearly acceptable value. The analysis of *H1* indicates that the Software Size is directly related to the Defect density. The percentage of defect density prediction by Size is very low because in the smaller size of software, it shows the declining trend with defect density [15].

Similar to the result of *H1*, the *H2* hypothesis *p*-value is also at the acceptable level. Thought this hypothesis is not rejected and the analysis result indicates that number of defects positively related to defect density. The percentage of defect density prediction by defects is more than by the size.

The analysis results for hypothesis *H3* shows acceptable *p*-value and Number of Developer predicts the significant percentage of Defect density. More number of developers of open source software increases the defect density. This approach establishes the positive relationship between Number of Developer and defect density, but it was opposed by Weyuker Alaine J. et al. [4].

On the basis of the analysis result for *H4*, it inferred that the Number of Download does not affect the defect density at the significant level. The R-squared value also indicates a very small percentage of prediction for defect density. Hence the hypothesis has to be rejected. The similar relationship shows by the study [10], but the significant improved result has been proposed in this paper.

Similar to hypothesis *H4*, result for *H5* is not at an acceptable level and shows a very weak relationship between Number of Commits and Defect density. The predicted percentage of defect density by Number of Commits is more than the Number of Downloads because the number of commits indicates the more changes in the code that may lead to newer defects in the code. This metric has not been considered before in relation to the defect density. Because the analysis results for this metric are not acceptable, hence the hypothesis has to be rejected.

In previous all five hypotheses, metrics show the impact on defect density independently. In hypothesis *H6*, a combined impact on defect density by all five metrics has been analyzed. The result shows the acceptable value and more percentage of defect density prediction by all five metrics jointly. In the work done by Rahmani C. et al. [10], they have only discussed the relationship between four repository metrics independently and they have only joined maximum two metrics for combined effect on defect density. The result of our proposed approach shows more improvement in prediction percentage of defect density by all metrics jointly.

7 Threats to Validity

Since this is an empirical study, there are some potential threats to the validity of the results that require a discussion:

- In this work, defect density was calculated using two measures: total number of defects in each project and, the size of the project itself. These two data have been taken from the CVS repository of project entered by the developers. Few issues may arise such as: some defects may not surface, some defects may surface, but not get fixed and, some defects may not be recorded in the repository. The community of users and developers of the projects used in this study is very large, so it may be confined that the defects were identified and fixed adequately.
- The programming language of the project affects the size of the project. In this study JAVA projects have filtered out for analysis amongst a very large number of projects on the SourceForge.net, considering most of the community uses projects with JAVA as a programming language.
- Some other factors such as developers' skills, working conditions, and understanding of the project may affect the complexity of the project as well as number of defects in the project. This type of information is not available on the project's web page.

8 Conclusions and Future Work

In this work, a relationship of defect density with different repository metrics of open source software has been established with the significance level. Five repository metrics namely Size of project, Number of defects, Number of developers, Number of downloads, and the Number of commits have been identified for predicting the defect density of open source project. This relationship can be used to predict the defect density of open source software. An analysis has been performed on 62 open source software available at sourceforge.net. Simple and multiple linear regression statistical methods have been used for analysis. The result reveals a statistically significant level of acceptance for prediction of defect density by some repository metrics individually and jointly. Further, other repository metrics can be combined to analyze the prediction. Some static metrics may also be used in combination of repository metrics to predict the defect density of open source software.

References

1. Trung T. D. & James M. B.(2005). The FreeBSD Project: A Replication Case Study of Open Source Development. IEEE transaction on software engineering, 31, 6, 481-494.
2. Sourceforge, http://sourceforge.net/. [Accessed: on April 19, 2015].
3. Chidamber S. & Kemerer C. (1994). A metrics suite for object oriented design. IEEE Transaction of software engineering, 20, 6, 476-493.
4. Weyuker E.J., Ostrand T.J., & Bell R. M. (2008). Do too many cooks spoil the broth? Using the number of developers to enhance defect predication models. Empirical software engineering, 13, 5, 539-559.
5. Gyimothy T., Ferenc R., & Siket I. (2005). Empirical validation of object oriented metrics on open source software for fault prediction. IEEE Transaction on software engineering, 31, 10, 897-910.
6. Knab P., Pinzger M., & Bernstein A. (2006). Predicting Defect Densities in source code files with Decision Tree Learners. Proceedings of the 2006 international workshop on Mining software Repositories, 119-125.
7. Subramanyam R. and Krishnan M.S. (2003). Empirical Analysis of CK Metrics for Object-Oriented Design Complexity: Implications for Software Defects. IEEE Transactions on Software Engineering, 29, 4, 297-310.
8. Mockus A., Fielding R.T., & Herbsleb J.D. (2002). Two case studies of open source software development: Apache and Monzilla. ACM Transaction on software engineering methodology, 11, 3, 309-346.
9. Sherriff M., Williams L., & Vouk M. (2004). Using In-Process Metrics to predict Defect Density in Haskell Programs. The 15th International Symposium on Software Reliability Engineering.
10. Rahmani C. and Khazanchi D. (2010). A Study on Defect Density of Open source software. The 9th International conference on Computer and Information Science, 679-683.
11. Caglyan B., Bener A., & Koch S. (2009). Merits of Using Repository Metrics in Defect Prediction for Open Source Projects. International Conference on Software Engineering, 31-36.
12. Park, R. (1992). Software Size Measurement: A Framework for Counting Source Statements. CMU/SEI-92-TR-20, Software Engineering Institute, Pittsburgh, PA.
13. Institute of Electrical and Electronics Engineers 1044-2009 (2010), "IEEE Standards Classification for Software Anamolies," Available at http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5399061.
14. Alali A., Kagdi H., & Maletic J.I. (2008). What's Typical Commit? A Characterization of Open Source Software Repository. Proceedings of 16th IEEE International Conference on Program Comprehension, 182-191.
15. Basili V. R. & Perricone B. T. (1982). Software Errors and Complexity: An Empirical Investigation. Communication of ACM, 27, 1, 42-54.
16. Statistical Package for Social Sciences (n.d.). Available at http://www.spss.com/statistics. IBM SPSS Statistics Version 20 64-bit.
17. Poole M.A. & O'Ferrell P.N. (1971). The assumptions of the linear regression model," Inst. Brit. Geogr., Trans., 52, 145–158.
18. Elliott A.C. & Woodward W.A. (2007). Statistical analysis quick reference guidebook with SPSS examples. 1st ed. London, Sage Publications.

19. Verma D. & Kumar S. (2014). An Improved Approach for Reduction of Defect Density Using Optimal Module Sizes. Hindwai Publishing Corporation, Advances in Software Engineering, Volume 2014, Article ID 803530.
20. Verma D. & Kumar S. (2015). Exponential Relationship Based Approach For Predictions Of Defect Density Using Optimal Module Sizes. Proceedings Of National Academy Of Sciences Section A: Physical Sciences, DOI 10.1007/s40010-015-0261-x.
21. Raza A., Capretz L.F., & Ahmed F. (2012). Users' perception of open source usability: an empirical study. Journal of Engineering and Computers, 28, 2, 109-121.
22. Keng S. & Yuhong T. (2013). Open Source Software Development Process Model: A Grounded Theory Approach. Journal of Global Information Management, 21, 4, 103-120.
23. Jiang Y., Li M., & Zhou Z. (2011). Software Defect Detection with Rocus. Journal of Computer Science and Technology, 26, 2, 328-342.
24. Verma D., Mandhan N., & Kumar S. (2015). Analysis of Approach for Predicting Software Defect Density using Static Metrics. International Conference on Computing, Communication and Automation, 880-886.
25. Deepak N. & Kumar S. (2015). Flexible Self-Managing Pipe-line Framework Reducing Development Risk to Improve Software Quality. International Journal of Information Technology and Computer Science, 7, 7, 35-47.