A Study on Defect Density of Open Source Software

Cobra Rahmani and Deepak Khazanchi, Ph.D.
College of Information Science and Technology
University of Nebraska-Omaha,
Omaha, U.S.
E-mail: {crahmani, khazanchi}@unomaha.edu

Abstract— **Open source software (OSS) development is considered an effective approach to ensuring acceptable levels of software quality. One facet of quality improvement involves the detection of potential relationship between defect density and other open source software metrics. This paper presents an empirical study of the relationship between defect density and download number, software size and developer number as three popular repository metrics. This relationship is explored by examining forty-four randomly selected open source software projects retrieved from SourceForge.net. By applying simple and multiple linear regression analysis, the results reveal a statistically significant relationship between defect density and number of developers and software size jointly. However, despite theoretical expectations, no significant relationship was found between defect density and number of downloads in OSS projects.**

Keywords- Bug repository; Defect density; Linear regression; Open source software; Software repository metric;

I. INTRODUCTION

Over the past years, Open Source Software (OSS) has drawn increasing attention, both from the business and academic world. Raymond [1] differentiates the traditional model of software development from the new collaborative model of open source software development. In terms of the large number of developers who cooperate and take advantage of parallel work in open source projects, this development style ultimately guides developers to rapid progress in open source software development. Also due to the nature of OSS, developers use the web as a shared distributed development forum for all aspects of software design and maintenance. Consequently, more people and companies get involved in the OSS development process. OSS products are used as development tools as well as embedded components of commercial projects in many software organizations and corporations [2].

OSS users act as developers as well as the test teams for open source projects. Since a certain development style and culture is implicit to OSS projects, no formal definition or description of an open source development process exists. Therefore there is considerable variance in the practices employed in these projects [3].

In this kind of development environment, people are acutely conscious of maintaining an acceptable level of quality and minimizing faults. In order to address the quality issue, a prediction of the defect density within an OSS project is a promising subject of exploration. By discovering the relationship between defect density and software repository metrics, erroneous OSS will be tracked down in the early phase of development when adapted by commercial projects. Repository metrics are those that simply can be extracted from software repositories.

Our fundamental research question is: "Is there a relationship between defect density of OSS projects and number of downloads, software size, and number of developers?"

However, prediction of defect density trend using software quality metrics like OO metrics defined by Chidamber and Kemerer [4] (e.g. Coupling Between Object classes, Number of Children, Depth of Inheritance Tree, Weighted Methods per Class, and etc.) needs lots of information from code detail. As an alternative, using repository metrics presents an easy and reliable method to assess defect density indirectly. Therefore, in this research we use repository metrics such as number of developers, number of downloads and software size, to predict the future of defect density in OSS.

By answering the research question, OSS users can potentially get a better idea about the quality of OSS just by looking at some disclosed information from the OSS projects ahead of OSS adoption.

To provide enough information from a wide range of OSS projects, we utilized forty-four randomly selected open source projects from SourceForge.net [5]. The projects are selected based on following additional considerations.

- The OSS products have an activity percentile of ninety five percent or more in SourceForge. An activity percentile of ninety-five percent means that the project is more active than ninety-five percent of all other projects on SourceForge during the given time period [6].
- Software size is calculated within the same operating environment - all OSS projects selected are Java-based and Windows compliant.
- Besides high activity percentile, the OSS projects with inactive bug repository or insufficient amount of failure data have been removed from study.

978-0-7695-4147-1/10 $26.00 © 2010 IEEE
DOI 10.1109/ICIS.2010.11

The remainder of this paper is organized as follows: Section II describes some of the related works and background knowledge used in this research; Section III outlines the research methodology and explains simple and multiple linear regression test that is used in this study, and the results achieved by this test; Section IV presents concluding remarks and summarizes the paper.

II. BACKGROUND

There is very limited research that has focused on defect density and its relationship with repository metrics in OSS projects. In their paper, "Do too many cooks spoil the broth?", Weyuker et al. used the number of developers as a variable to enhance defect prediction models [7]. They investigated the effect of number of developers in one large-scale closed source project. They found that the numbers of developers in commercial projects do not have any effect on the defect density. Mockus et al. [8] report on their research on predicting the defect density in Apache and Mozilla projects. They based their conclusions on two open source projects and the authors did not assess the number of developers as a distinguishing factor. Additionally, Koch and Neumann [9] used coordination and communication measures in a study of open source frameworks. They found that the use of a high number of programmers is associated with problems in quality on class level but they did not consider the effect of other metrics like number of downloads on defect density.

In [10], authors present a quantitative model to predict the value of defect density by using some static metrics. Although these static code attributes are useful for modeling software data in defect prediction, but relying on just code information to present this model is not enough. Since software is a complex system with many possible approaches for abstracting it, there is a need to explore alternative metric sets such as repository or organizational metrics for defect prediction [3]. We propose one possible solution to this gap in our understanding by exploring the use of repository metrics such as number of downloads and developers in addition to software size, to predict the future of defect density. This can potentially lead to better estimation of OSS quality.

A. Conceptual definitions

Before presenting our research analysis, we provide some conceptual descriptions of the key constructs in the study.

Software size - Either size can be counted in Lines of Code (LOC) or in the number of function points. Between these two factors, lines of code easily can be calculated for entire software product. Several variations of LOC is presented in [11] as follows:

a) Only executable lines

b) Executable lines plus data definitions

c) Executable lines, data definition and comments

d) Executable lines, data definition, comments, and job control language

e) Physical lines on an input screen

f) Lines as terminated by logical delimiters

A java program has been used to count all physical lines of OSS projects included in our study.

Defect - Based on IEEE standard [12], a defect is an anomaly in a product and a failure occurs when a functional unit of a software-related system can no longer perform its required function or cannot perform it within specified limits. When a defect happens, it can cause several failure situations or do not appear as a failure as long as the product has not been executed for a long time with a special scenario [13].

The number of known defects, which is used to calculate defect density, is the count of total defects identified against a particular software entity during a particular time period. Examples include [14]:

a) Defects to date since the creation of module

b) Defects found in a program during an inspection

c) Defects to date since the shipment of a release to the customer

Defect density - Defect density is a measure of the total known defects divided by the size of the software entity being measured [14].

$$Defect\ density = \frac{Number\ \ of\ known\ \ defects}{Software\ \ size}. \quad (1)$$

To define the defect density, the numerator and denominator have been specified as discussed above. Due to the lack of data to calculate the actual number of defects in software, number of failures, which is reported by customers, is considered as an approximate measure of known defects in the software. In this study, defects are counted to date since the release of OSS in SourceForge.net.

Number of downloads – This is the number of downloads of the project, as given by the respective statistics page in SourceForge.net at the time of collecting data.

Number of Developers - This is the number of developers as reported by the project's page on SourceForge.net.

III. RESEARCH ANALYSIS

As discussed before, the overall research question in this work is to study the relationship between defect density and the three repository metrics, i.e. number of downloads, software size, and number of developers.

Actual names of the projects are not revealed to follow standard software engineering ethics [15]. We calculate defect density using the following formula.

$$Defect\ density = \frac{Total\ \ Number\ \ of\ known\ \ defects}{KLOC}, \quad (2)$$

where KLOC = LOC*1000.

Table I displays the hypotheses to address the research questions of this study.

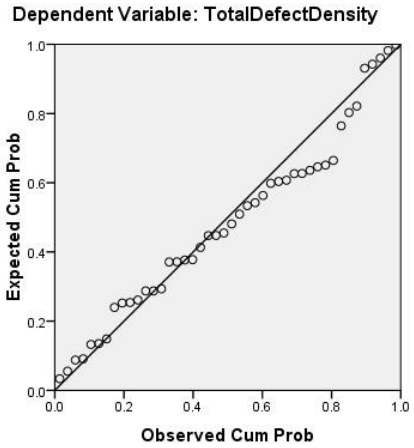
TABLE I. RESEARCH HYPOTHESES AND THEIR RESULTS

HypId	Hypothesis Text	Results after study
H1	H01: High number of downloads does not have any relationship with defect density in OSS.	Not Rejected
	HA1: High Number of downloads increases defect density of OSS.	Not Rejected
H2	H02: More number of developers in OSS project is not related to defect density.	Rejected
	HA2: OSS projects with more number of developers have higher defect density.	Accepted
H3	H03: Software size is not correlated with defect density of OSS.	Not Rejected
	HA3: Software size in OSS has a relation with defect density.	Not Rejected
H4	H04: Number of developers and software size jointly are not related to defect density of OSS.	Rejected
	HA4: Number of developers, and software size jointly have a relationship with defect density of OSS.	Accepted

Using SPSS [16], linear regression analysis is applied to determine whether the dependent variable, i.e. defect density, varies as a function of one or more independent variables or not.

The regression tool in SPSS offers many statistical options such as the Histogram plot, Analysis of Variance, R-squared, and the significance of the observed regression line (P-value). P-value is a significance level to reject or accept the statistical tests. The p-value represents the probability of error that is involved in accepting our observed result as valid, that is as "representative of the population." For example, a p-value of .05 indicates that there is a 5% probability that the relationship between the variables under consideration is a "fluke" [17]. The ANOVA tool in SPSS gives a F statistic for testing the claim that there is no significant relationship between an independent variable and dependent variables. One output for this test is the p-value. Thus we can reject the claim that there is no significant relationship between the independent and dependent variables if p-value is smaller than α. In most statistical tests the value which is used for α can be 0.01, 0.05, or 0.1 and we selected 0.01 for these statistical tests. We calculate the p-values to decide whether to reject the null hypothesis (H0) or not.

Before doing each test, the samples are analyzed for normality, and the assumption of equality of variances. Hair et al.[18] suggest that when a study has a small sample size, it is safer to use both a normal probability plot and test statistics to ensure the normality. The Kolmogorov-Smirnov (Lilliefors) and Shapiro-Wilks normality tests passed for all of the hypotheses tested in our study. In addition, the normal probability plot for each variable indicates a normal distribution of the sample. Because of the limitation in terms of paper lenght, only the normality output for hypothesis H4 is shown in the paper. As shown in Figure 1, residuals of errors of multiple linear regression are normally distributed and both tests are non-significant (α= 0.01). The normal probability plot is also shown in Figure 1.



(1.a)

Tests of Normality

	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Statistic	*df*	*Sig.*	*Statistic*	*df*	*Sig.*
Standardized Residuals	.143	44	.024	.957	.44	.103

a. Lilliefors Significance Correction

(1.b)

Figure 1. a) Normal probability plot, b) K-S and S-W results for H4

The R-squared statistics describes how much of the variation of the independent variable is explained by the variation of the dependent variables [19]. It is up to researcher to interpret the results based on R-squared value. An R-squared value closer to 1.0 indicates that most of the variation in the dependent variable is explained by the variations in the independent variables.

In this research, best values calculated for R-squared are around 0.35, which may not be an attractive number, but shows that thirty five percent of the variation of the defect density can be explained by independent variables which is a promising number for repository metrics.

A. H1: Download numbers are positively correlated with defect density

In this section, we evaluated whether defect density is correlated with the number of downloads in OSS or not. Figure 2 presents a plot with number of downloads on the x-

axis and defect density on the y-axis. Each point in the plot presents one software. This plot shows the regression linear line function with the appropriate R-squared value.

TABLE II. R-SQUARED AND P-VALUE FOR DEFECT DENSITY AGAINST NUMBER OF DOWNLOADS

R-squared	0.001
p-value	0.862

As shown in the Table II, the results indicate that there is no linear relationship between defect density and the number of downloads. Since number of downloads implicitly represents the number of users for OSS, this study does not support a commonly held view previously presented by Raymond, "with enough eye balls, all bugs are shallow" [1].

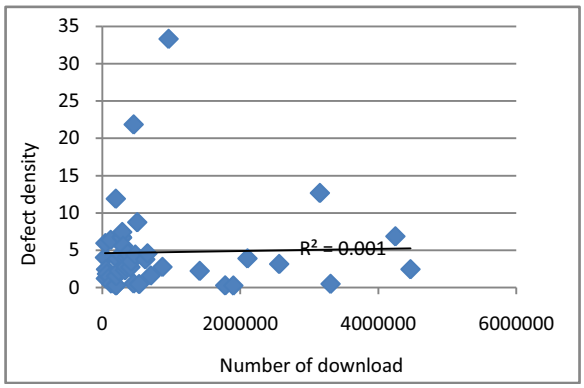


Figure 2. Regression analysis between defect density and number of downloads

B. *H2: The number of developers are positively correlated with defect density*

In this hypothesis, we are particularly interested in knowing whether increasing the number of developers on an OSS project has any effect on the number of defects in a specified software size or not.

The statistcal results for this hypothesis are presented in Table III and the relevant plot is displayed in Figure 3. Based on the calculated p-value which is shown in Table III, the hypothesis that number of developers are positively correlated is suported. Although R-squared is low, the p-value indicates that there is a significant relationship between number of developers and defect density. The low R-squared number potentially explains the fact that the developer number is not the only predictor for defect density. Other predictors in conjunction with this variable would better determine the amount of defects in OSS projects.

TABLE III. R-SQUARED AND P-VALUE FOR DEFECT DENSITY AGAINST NUMBER OF DEVELOPERS

R-squared	0.259
p-value	0.000

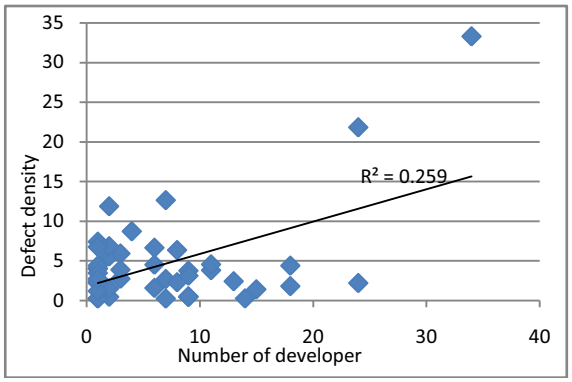


Figure 3. Regression analysis between defect density and number of developers

In this research, only three repository metrics separatly and jointly are analyzed to discover their effect on defect density. More researches are needed to discover the potential relationship with other software metrics such as object oriented metrics on software defect.

C. *H3: The software size is positively correlated with defect density*

In this hypothesis we made an effort to find the potential relationship between project size measured in terms of KLOC and defect density. Early studies of this relatinship pointed to a negative relationship, i.e., the larger the project size, the smaller the defect rate. For example Basili and Perricone [20] studied Fortran projects with less than 200 LOC and found more defect density in the smaller projects. Similarly, Shen and et al. [21] studied projects written in Pascal, PL/S, and Assembly language and showed an inverse relation existed up to about 500 lines of code. They concluded that the larger a module, the smaller its potential defect density. More recent studies point to a curvilinear relationship between lines of code and defect density, i.e., defect density decreases with size and then curves up again at the tail when the modules become very large [13].

In our analysis of OSS project, we try to first explore for a linear relationship between Defect density and KLOC in Java-based OSS projects. As shown in Table IV, the R-squared and p-value confirms that there is no statistcailly significant linear relationship.

TABLE IV. R-SQUARED AND P-VALUE FOR DEFECT DENSITY AND KLOC

R-squared	0.031
p-value	0.248

D. *H4: A multiple regression model of defect density and repository metrics*

After examining several simple regression analyses of the data, we were interested in studying multiple regression analysis model. Since multiple regression analysis studies the relationship of independent variables jointly, the results can potentially be more accurate.

In this step, one multiple regressions with all of those three predicators, i.e. software size, number of developers, and number of downloads has been performed. Furthermore, several different combinations of two predictors are analyzed. In terms of p-value and R-squared the results with all three predictors do not show any improvement over two predictors of software size and number of developers. The relationship between defect density, on one hand, and OSS size and number of developers on the other, is the best amongst all combinations. Table V shows p-value and R-squared calculated for this case.

TABLE V. R-SQUARED AND P-VALUE FOR DEFECT DENSITY AGAINST KLOC AND NUMBER OF DEVELOPERS

R-squared	0.352
p-value	0.000

The p-value in this case is less than 0.01, which means KLOC and developer numbers jointly are good predictors of defect density. In addition, the R-squared value is better than all the previous cases. This means that the number of developers and software size together present the strongest relationship with defect density.

IV. SUMMARY AND CONCLUSION

In this paper we address the research question - "is there a relationship between defect density of OSS projects and number of downloads, software size, and number of developers?" Contrary to theoretical expectations, regression analysis discovered no significant relationship between defect density and number of downloads in OSS projects. However, the number of developers and software project size together present greater promise in explaining defect density of OSS projects. Further work in this area will extend the study to explore other potential predictors for defect density in OSS projects. Another potential avenue for future research is the use of non-linear regression to explain the trends in defect density associated with OSS project.

ACKNOWLEDGEMENT

This research is funded in part by Department of Defense (DoD)/Air Force Office of Scientific Research (AFOSR), NSF Award Number FA9550-07-1-0499, under the title "High Assurance Software".

REFERENCES

[1] E. S. Raymond, "The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental evolutionary," Cambridge: O'reilly and Associates, 1999.

[2] J. Rudzki, K. Kiviluoma, T. Poikonen, and I. Hammouda, "Evaluating Quality of Open Source Components for Reuse-Intensive Commercial Solutions," 35th Euromicro Conference on Software Engineering and Advanced Applications, August 27- 29, 2009, Patras, Greece.

[3] B. Çaglayan, A. Bener, S. Koch, "Merits of Using Repository Metrics in Defect Prediction for Open Source Projects," ICSE'09 Workshop, FLOSS'09, May 18, 2009, Vancouver, Canada.

[4] S. Chidamber and C. Kemerer, "A metrics suite for object-oriented design," IEEE Trans. Software Eng., vol. 20(6), pp. 476-493, 1994.

[5] "Sourceforge" , http://sourceforge.net/. [Accessed: Jan. 14, 2010].

[6] B. Cornelius, S. Rahtz, S. Yeates, R. Gardler, "Finding your way around SourceForge," Available: http://www.oss-watch.ac.uk/resources/sfintro.xml. [Accessed: March. 10, 2010].

[7] E. Weyuker, T. Ostrand, and R. Bell, "Do too many cooks spoil the broth? using the number of developers to enhance defect prediction models," Empirical Software Engineering, vol. 13(5), pp. 539–559, 2008.

[8] A. Mockus, R. Fielding, and J. D. Herbsleb, "Two case studies of open source software development: Apache and Mozilla," ACM Trans. Softw. Eng. Methodol., vol. 11(3), pp. 309–346, 2002.

[9] S. Koch and C. Neumann, "Exploring the effects of sourceforge. net coordination and communication tools on the efficiency of open source projects using data envelopment analysis," Empirical Software Eng vol. 14, pp. 397–417, 2009.

[10] M. Sherriff, L. Williams, and M. Vouk, "Using In-Process Metrics to Predict Defect Density in Haskell Programs," Fast Abstract. The 15th International Symposium on Software Reliability Engineering, St-Malo, France, Nov 2-5, 2004.

[11] Jones, C., Programming Productivity, New York: McGraw-Hill, 1986.

[12] "IEEE standard 982.2, Guide for the Use of IEEE Standard Dictionary of Measures to Produce Reliable Software," Available: http://standards.ieee.org/reading/ieee/std_public/description/se/982.2-1988_desc.html. [Accessed: March. 22, 2010].

[13] Kan H.S., Metrics and Models in Software Quality Engineering, 2nd ed. Edition, MA: Addison-Wesley, 2003.

[14] L. Westfall, "Defect Density," Available: http://www.westfallteam.com/Papers/defect_density.pdf. [Accessed: Feb. 2, 2010].

[15] K. El-Emam, "Ethics and Open Source," Empirical Software Engineering, vol. 6, issue 4, pp. 291-292, Dec. 2001.

[16] "SPSS", http://www.spss.com/statistics. [Accessed: March. 22, 2010].

[17] "StatSoft Electronic statistics textbook," Available: http://www.statsoft.com/textbook/elementary-concepts-in-statistics/ [Accessed Apr. 1, 2010].

[18] J. F. Hair, R. E. Anderson, R. L. Tatham and W. C. Black, Multivariate Data Analysis. 3rd ed., New York: Macmillan, 1992.

[19] P. Mohagheghi, R. Conradi, O. M. Killi, H. Schwarz, "An empirical study of software reuse vs. defect-density and stability," 26th International Conference on Software Engineering, May 23-28, 2004, Scotland, UK.

[20] V.R. Basili, and B. T. Perricone, "Software Errors and Complexity: An Empirical Investigation," Communications of the ACM, vol. 27, no. 1, pp. 42-54, Jan. 1984.

[21] V. Shen, T. Yu., S. Thebaut, and L. Paulsen, "Identifying error prone software – an empirical study," IEEE Transaction in Software Engineering, vol. 11, no. 4, pp. 317-324, Apr. 1985.