Hindawi Publishing Corporation
ISRN Software Engineering
Volume 2013, Article ID 198937, 18 pages
http://dx.doi.org/10.1155/2013/198937

Research Article

An Empirical Study of the Effect of Power Law Distribution on the Interpretation of OO Metrics

Raed Shatnawi and Qutaibah Althebyan

Software Engineering Department, Jordan University of Science and Technology, Irbid 22110, Jordan

Correspondence should be addressed to Raed Shatnawi; raedamin@just.edu.jo

Received 21 November 2012; Accepted 20 December 2012

Academic Editors: P. Ciancarini, J. A. Holgado-Terriza, and Z. Shen

Context. Software metrics are surrogates of software quality. Software metrics can be used to find possible problems or chances for improvements in software quality. However, software metrics are numbers that are not easy to interpret. Previous analysis of software metrics has shown fat tails in the distribution. The skewness and fat tails of such data are properties of many statistical distributions and more importantly the phenomena of the power law. These statistical properties affect the interpretation of software quality metrics. *Objectives*. The objective of this research is to validate the effect of power laws on the interpretation of software metrics. *Method*. To investigate the effect of power law properties on software quality, we study five open-source systems to investigate the distribution and their effect on fault prediction models. *Results*. Study shows that power law behavior has an effect on the interpretation and usage of software metrics and in particular the CK metrics. Many metrics have shown a power law behavior. Threshold values are derived from the properties of the power law distribution when applied to open-source systems. *Conclusion*. The properties of a power law distribution can be effective in improving the fault-proneness models by setting reasonable threshold values.

1. Introduction

Software measurement includes collecting data about properties of software classes to verify the quality of software using internal properties such as size, coupling, and inheritance. Software designers can adapt quality assurance tools to measure the quality of software properties and analyze them using graphical analyses such as histogram analysis, box plots, and correlation matrix. The analyses of metrics data in many previous works [1–3] have shown, using histograms or descriptive statistics, that metrics are right skewed. This skewness in metrics data affects the interpretation and usage of these metrics in evaluating software quality. Such metrics are not always well characterized by their descriptive statistics such as mean, standard deviation, minimum, maximum, and quartiles.

Software metrics have been used as indicators for software quality such as software fault proneness and maintenance effort [1, 2, 4–12]. In these studies, usually large values were found correlated with a large number of defects. In other studies, large values were found to correlate with bad design and errors [13–15]. Although there are immense research studies on validating the software metrics, few have done work on the interpretation of software metrics. For example, Rosenberg has suggested analyzing the relationship between metrics and software quality using histograms [16] and has suggested a set of threshold values for some metrics; these values can be used to select classes for inspection or redesign [16, 17]. In another study, Erni and Lewerentz [18] proposed a technique to identify metric threshold values based upon the mean and the standard deviation of software metrics. Shatnawi [19] has proposed another method based on the distribution of data to select a threshold value for a particular metric by analyzing misclassification costs of the modules that were classified into either the error or no-error groups. The method has used distribution parameters to find threshold values that classify software modules [19].

Data distribution affects the interpretation and application of the metrics in practice. There are many studies on the distribution of metrics. Nevertheless, power law distribution

is a phenomenon that has been found in many object-oriented properties. A power law describes a common behavior which states that "*there are few very complex modules while most modules have low complexity.*" However, if a software metric follows a power law distribution, then a power law distribution may not have a finite mean and variance, and so the central limit theorem cannot be utilized to set threshold values as have been suggested in previous works [16–19]. Louridas et al. [20] reported that the observation of the power law distribution on static metrics has an impact on several aspects of software engineering. The observation of power law helps in allocating resources efficiently during software development, that is, allocating resources unequally among the software parts. In addition, they have suggested using the power law to identify the most reusable components the fault-prone components, and to optimize the efficiency of software applications. Wheeldon and Counsell [21] have stated that "A power law implies that smaller values are extremely common, whereas larger values are extremely rare." The power laws can be used to learn the likely features of classes, in and parameters of the power law distribution can be used (the exponent) to compare between the metric values for different systems [21]. For example, a small exponent is an indicator of less skewed data, whereas a large exponent may suggest that the software has rare key classes (large classes) [21]. Such criterion can be used to set threshold values for software metrics or to estimate the maximum value of a particular metric. Therefore, The objectives of this research are as follows.

(i) To explore whether software metrics follow the power law distribution or not.

(ii) To find the effect of the power law on metric threshold values. We want to find whether there are bounds for a particular metric.

(iii) To study the evolution of software systems using the power law characteristics in evolutionary designs.

(iv) To study the effect of the power law characteristics on the fault prediction models.

The rest of the paper is structured as follows: Section 2 is dedicated to the related work of fitting metrics to a power law; Section 3 presents the research methodology and data collection; Section 4 introduces the power law distribution and its characteristics and a maximum likelihood estimation of the power law estimates; and in Section 5 we analyze the data. And finally, we use the properties of the power law to build fault-proneness models.

2. Related Work

The power law distribution is a phenomenon that has been found to characterize many scientific data such as physics, biology, earth sciences, economics, and computer science [22]. Recently, many research studies were conducted to show whether software metrics follow a power law distribution or not. Valverde et al. [23, 24] studied the power law in large Java and C/C++ open-source software systems at the design level and found that nodes in class diagrams follow a power law distribution. Wheeldon and Counsell [21] found that twelve static metrics follow a power law distribution when fitted using linear regression on the log-log data plots. Baxter et al. conducted a more comprehensive study on seventeen static metrics and found that some metrics follow a power law when fitted using weighted least squares, while others can be described by other distributions [25, 26]. Concas et al. [25, 26] studied many system properties of the implementation of a Smalltalk system (Visual Works Smalltalk). They validated whether the data follow a log-normal or a Pareto distribution using maximum likelihood estimations. They have found that most studied metrics followed a power law distribution except for the number of methods metric (i.e., the WMC metric), the number of all instance variables metric, and the CBO metric, which follow a log-normal distribution. Louridas et al. [20] studied the evidence of power law distribution in many software systems at two levels: class and function levels. In addition, they showed that the power law distributions appear at various levels of abstraction, platforms, and languages. Louridas et al. measured the software complexity using two static metrics, the fan-in and the fan-out metrics, for each module (a class in Java or a function in C, Perl, or Ruby) [20]. They used the least square estimate (r^2) to benchmark the fit to a power law. The evidence of power law distribution has suggested that software engineers can focus on optimal dependencies between modules, while avoiding the parts of a software architecture that is highly connected (i.e., high fan in or fan out). Hatton studied whether component sizes, measured as SLOC, obey a power law distribution or not [27]. He found, on 21 software systems implemented in three different languages (C, Fortran, and TCL) for various functionalities, that the SLOC obeys a power law distribution using a log-log plot of rank frequency for the size of each system (i.e., at the microscopic level) and for the size of all systems combined (i.e., at the macroscopic level). In another study, Hatton studied the persistence of power law behavior on four software systems under development for many releases starting from the first release [27]. He noticed, in all studied releases, that there has been no substantial change in the shape of components and concluded that the power law behaviour is persistent.

The Pareto principle has been under investigation in many studies as well. Andersson and Runeson [28] replicated a previous study on fault distribution of complex software systems that was conducted by [10]. Anderson and Runeson quantitatively analyzed the fault distribution in three different projects. They tested whether fault distributions, prerelease and postrelease faults, follow a Pareto rule (i.e., 20 : 80 rule) or not. They found by the means of a graphical analysis that a small number of modules (20%) contain a large number of prerelease faults (between 63% and 70%), and at the same time they contain a large number of postrelease faults (80%–87%). Andersson and Runeson [28] also reported results of some previous work for Fenton and Ohlsson [10] where they found (20 : 60) for prerelease faults and (10 : 80) for postrelease faults. Mubarak et al. [29] conducted a study

to explore whether the 20 : 80 rule or the Pareto principle exists in four Java systems for six coupling metrics over multiple versions. Mubarak et al. [29] found that one metric appeared to follow a Pareto principle, which was the fan-in metric. They did not use a fitting model; rather they used the rank-size plot to explore the existence of the rule. Other researchers found different results such as (38 : 80) in [30], (20 : 62) in [31] and (20 : 65) in [32]. In these previous works, the methodology of fitting to a power law has been discussed. There are two widely used methods to find a fitting of power laws: the visual methods using the linear fit of the log-log plot and the linear fitting using the least squares regression. Goldstein et al. [33] used a simple experiment to show that fitting to a power law distribution using graphical methods based on linear fit on the log-log scale is biased and inaccurate. They also have shown that the maximum likelihood estimation (MLE) is far more robust. In addition, log-normal distribution can be an alternative for the power law distribution. Mitzenmacher [34] found that log-normal and power law distributions are connected quite naturally, and hence it is not surprising to state that log-normal distributions could be a possible alternative to power law distributions across many fields including biology, chemistry, ecology, astronomy, and information theory.

In this work, we focus on (1) fitting OO metrics to a power law distribution using maximum likelihood estimation and (2) proposing a methodology that tests whether a power law distribution can be ruled out as a fit to software metrics, which can give more information about metrics interpretation.

3. Research Methodology

Software systems are composed of many interacting entities (classes, methods, and attributes) that constitute structures of systems. Structures of systems can be characterized using properties of these entities such as dependencies among classes, inheritance hierarchy, number of methods, and number of attributes. These properties can be used to measure the quality of software systems. From previous research, structures of systems were found to follow a scale-free behavior and, therefore, a power law distribution. This emerges from the fact that a software process includes a sequence of decisions that can be modeled as a random process [25, 26]. Our methodology includes collecting data for many open-source projects developed in Java. For these projects, we collect object-oriented and some well-known size metrics to study the effect of power law distribution on software systems. We also provide details of systems under study, collected metrics, and the descriptive statistics that characterize the metrics' data.

3.1. Data Collection. To provide a solid evidence of the effect of power laws, many software systems need to be investigated. Five open-source systems developed in Java were chosen, and for one of these systems we study the persistence of power law distributions over many releases (12 releases of JFreeChart system). The purpose of collecting the data for many releases is to investigate the effect of software evolution on the power law distribution if exists. Table 1 shows a description of these systems with their sizes. Brief descriptions of these systems are provided in Appendix A.

Table 1: Systems under study.

	No. of releases	No. of classes	Size (SLOC)
jBPM 4.0.CR1	1	684	24 K
OpenBravo ERP 2.50	1	907	123 K
Jedit 4.3pre16	1	1210	122 K
Eclipse	3	10898	1937 K
JFreeChart	12	1028	125 k

3.2. Software Metrics. In this research, we validate our objectives on the metrics that characterize the structure of object-oriented software systems, in particular the well-known Chidamber and Kemerer metrics which are known as the CK metrics [4]. This suite measures six software properties: coupling among objects (CBO), class responsibility (RFC), the inheritance structure (NOC and DIT), cohesion (LCOM), and weighted complexity (WMC). We exclude the LCOM metric from our investigation because of the criticism that this metric has received in previous works. Basili et al. [1] and Briand et al. [8] noted some problems in the definition of the LCOM metric (i.e., the LCOM metric gives the same value for classes with different cohesions). Etzkorn et al. [35] also reported similar conclusions about LCOM. In addition to the CK metrics, we include some of the well-known procedural metrics that characterize the size of a system such as SLOC, number of methods, and number of variables (data members) in a class. These metrics were previously found correlated with many software quality factors such as fault proneness of classes and maintenance effort [1, 3, 11, 36]. To collect these metrics, we use a commercial tool, Understand 2.0 (http://www.scitools.com/), which is specialized in evaluating software quality. In the following, brief descriptions of these metrics and their impact on software quality are presented as follows.

(i) Coupling between objects (CBO): the CBO metric counts the number of other classes to which the investigated class is coupled with. CBO measures the interconnections among classes.

(ii) Depth of inheritance tree (DIT): the DIT metric counts the number of ancestors of a class. The DIT metric connects chains of classes and specifies the complexity of changing classes deep in the inheritance hierarchy.

(iii) Number of children (NOC): the NOC metric counts the number of direct descendants of a class. The class that has many descendents is not easy to change and error prone.

(iv) Response set for a class (RFC): the RFC metric counts the number of methods in the response set for a class. This includes the number of methods in the class and the number of inherited methods in the class. Too

many responsibilities for one class is a sign of God classes; therefore, such classes should be divided to improve the modularity of the system.

(v) Weighted methods complexity (WMC): the WMC metric is the sum of the complexity of all methods in a class. Many metrics tools calculate the WMC metric as simply the number of methods in a class. This is equivalent to saying that all functions have the same complexity. However, in this research, the tool we used (Understand 2.0) calculates the WMC metric by summing the McCabe cyclomatic complexity of all the methods in a class. The WMC metric can be used to measure the testability of software systems.

(vi) Source lines of code (SLOC): the SLOC metric is the total number of lines of code in a class.

(vii) Number of variables (NOVs): the NOV is the number of variables in a class.

(viii) Number of methods (NOMs): the NOM is the number of methods in a class.

3.3. Power Laws. The power law distribution has many characteristics such as fat-tail distribution (i.e., right-skewed distribution) and scale-free networks (i.e., the existence of hubs in data where some classes abide most complexity). A scale-free behavior entails that the proportion of small to large values is preserved whenever systems evolve Wheeldon and Counsell [21]. Although a tree-like structure is expected for software systems [23], a quantity x follows a power law distribution if it is drawn from $P(x)$, where

$$P(x) = Cx^{-\alpha}. \tag{1}$$

The constant α is called the exponent of the power law (the constant C is mostly uninteresting and determined by the requirement that the $P(x)$ integrates to 1). The constant α usually falls in the range between 2 and 3. This constant can be estimated in various ways such as the least square fitting (after taking the log of the two sides and α becomes the slope of the line) and the method of maximum likelihood (MLE). Clauset et al. [37] have found that MLE produces estimates of α more accurate than those of the least squares. In another study, Goldstein et al. [33] have shown that using MLE is more robust than using graphical methods that were based on a linear fit on the log-log scale. Therefore, in this research, the MLE is considered to estimate the exponent of the power law. Although all metrics under study are discrete variables, the MLE provides calculations for both types of data, discrete and continuous variables. The estimation procedure of the exponent of a power law was adopted from two sources, Newman [22] and Clauset et al. [37]. The power law exponent using MLE for the continuous case is

$$\alpha = 1 + n\left[\sum_{i=1}^{n} \ln \frac{x_i}{x_{\min}}\right]^{-1}, \tag{2}$$

where x_i, $i = 1 \cdots n$, are the observed values of x such that $x_i \geq x_{\min}$. The power law exponent using MLE for the discrete case is

$$\alpha = 1 + n\left[\sum_{i=1}^{n} \ln \frac{x_i}{x_{\min} - 1/2}\right]^{-1}, \tag{3}$$

where x_i, $i = 1 \cdots n$, are the observed values of x such that $x_i \geq x_{\min}$, where the tail starts. It is very important, however, to test the hypothesis that x is drawn from a power law distribution for $x \geq x_{\min}$. The value of $x_{\min}$ should be estimated to have a goodness of fit to a power law distribution. If a too low value is chosen, then a biased parameter will be obtained for the model since the fitting will be to nonpower law data. On the other hand, if the chosen $x_{\min}$ is too high, then legitimate data points lower than $x_{\min}$ are excluded. Therefore, we need an estimation method that considers both constraints. A $x_{\min}$ value is chosen to minimize the differences between the probability distributions of the measured data and the best fit model. A statistical test such as the Kolmogorov-Smirnov (KS) goodness-of-fit test is used [38]. KS measures the distance between two probability distributions. The KS produces reasonable results when the data is right skewed. KS statistic is denoted D and is calculated as follows [37]:

$$D = \max |S(x) - P(x)|. \tag{4}$$

The best estimate of $x_{\min}$ is the one that minimizes D. $S(x)$ is the CDF (CDF denotes the cumulative distribution function) of the data for the observations with a value at least $x_{\min}$, and $P(x)$ is the CDF for the best fitted model in the region $x \geq x_{\min}$. Estimating these parameters is not sufficient to conclude that a given data is really drawn from a power law distribution for $x \geq x_{\min}$. Therefore, a goodness-of-fit test is needed to validate the hypothesis whether data follows a power law distribution or not [37]. A P value that is less than 0.1 is used to rule out the power law distribution. However, a P value larger than 0.1 does not indicate that the power law is the best fit, but we cannot rule out the power law as a plausible fit to the data.

3.4. The Characteristics of the Power Law Distribution. Data distribution is characterized using the estimated parameters such as the mean and standard deviation. In this section, the characteristics of a power law distribution are provided. The characteristics include how the mean, the standard deviation and the maximum value can be estimated for a metric. The estimated mean value of x in a power law distribution is given by [22]

$$\begin{aligned} \langle x \rangle &= \int_{X\min}^{\infty} x p(x)\, dx = C \int_{X\min}^{\infty} X^{-\alpha+1} dx \\ &= \frac{C}{2-\alpha}\left[x^{-\alpha+2}\right]_{X\min}^{\infty}. \end{aligned} \tag{5}$$

The mean becomes infinite if $\alpha \leq 2$. The power law distribution with such low values of α has no finite mean. This characteristic indicates that the maximum value increases for larger data sets and it is not upper bounded. The divergence

Table 2: Descriptive statistics for jBPM system.

	CBO	NOC	DIT	RFC	WMC	NOM	NOV	SLOC
Mean	3.5	0.8	1.9	11.8	8.5	.19	0.7	35.8
Median	2	0	2	7	4	.00	1	18
Mode	0	0	1	3	2	0	0	13
Standard dev.	4.3	3.3	1.2	15.2	15.1	1.584	1.1	60.7
Kurtosis	32.4	120.8	−0.1	31.7	69.6	*244.2*	27.3	58.3
Skewness	4.2	10.0	0.6	4.4	6.7	14.886	4.1	6.2
Minimum	0	0	0	0	0	0	0	0
Maximum	48	49	6	164	223	28	11	849

of (5) tells that as data gets larger, then the estimates of the mean will increase as well. Whereas, for $\alpha > 2$, the estimated mean converges (i.e., finite) and can be given by (6) [22]. This estimated mean depends on the exponent of the power law and the minimum value for which the power law behavior holds,

$$\langle x \rangle = \frac{\alpha - 1}{\alpha - 2} x_{\min}. \tag{6}$$

The mean square is calculated as in (7), which diverges if $\alpha \leq 3$. Thus, power law distributions of such values have no finite standard deviation [22]. The consequence of having infinite mean or standard deviation is that the central limit theorem does not hold for such distributions, thus the mean and variance of a sample (which will always be finite) cannot be used as estimators for the population mean and variance [25, 26],

$$\left\langle x^2 \right\rangle = \frac{C}{3 - \alpha} \left[x^{-\alpha+3} \right]_{X\min}^{\infty}. \tag{7}$$

For $\alpha > 3$, the mean square is finite and is given by [22]

$$\left\langle x^2 \right\rangle = \frac{\alpha - 1}{\alpha - 3} x_{\min}^2. \tag{8}$$

If $2 < \alpha \leq 3$, the mean of the distribution converges, but its standard deviation diverges as new measurements are added. This fact happens commonly in very fat tails. If $\alpha > 3$, both the mean and the standard deviation of the distribution converge [2]. In addition, extreme values depend on the number of entities and the value of the tail index (α). The maximum value that can be found in N samples is defined in (9). Therefore, the maximum value is size dependent, and the application size affects the bounds of the power law,

$$\langle x_{\max} \rangle = x_{\min} N^{1/(\alpha-1)}. \tag{9}$$

4. Results Analysis

The analysis has two steps: first, we analyze the data using descriptive statistics and second, we fit the metrics to the power law.

4.1. Descriptive Statistics. A normally distributed data is strongly centered on their mean values, hence the mean is representative for most observations. However, some data is skewed, which means that the data may not follow a normal distribution and the data is not distributed evenly around the mean.

One way to understand the distribution of large data sets is to use visual representations like scattered diagrams or histograms [39]. Graphical methods, although appealing, do not provide objective criteria to determine the normality of variables. Interpretations are thus a matter of expert judgments. Numerical methods use descriptive statistics, such as skewness and kurtosis, to examine normality. Skewness measures the degree of symmetry of a probability distribution. If skewness is greater than zero, the distribution is skewed to the right, having more observations on the left. Kurtosis measures the thinness of tails or peakedness of a probability distribution. Positive kurtosis indicates a relatively peaked distribution. Negative kurtosis indicates a relatively flat distribution. Normally distributed random variable should have skewness and kurtosis both near zero [40]. Skewed data can also be discovered by observing high values of the standard deviation with respect to the mean and the median. In skewed data, the maximum values are much larger than the mean; such data has a fat-tail property. Skewed data often occurs due to lower or upper bounds on the data which helps in deciding the direction of threshold values. That is, data that has lower bounds is often skewed right while data that has upper bounds is often skewed left. Skewness can also result from start-up effects. For example, in reliability applications, some processes may have a large number of initial failures that could cause left skewness [41]. In previous research on software metrics, it has been noticed that many descriptive statistics and graphs of the metrics' data observe right skewness such as in [1, 12]. In this section, the descriptive statistics for all systems under consideration are provided. In Tables 2, 3, 4, 5, and 6, it is observed that mean ≥ median ≥ mode for all metrics, which is obviously a sign of right skewness in the data. This behavior is caused by few large classes in the system, and it can be noticed that the maximum values are much larger than the mean and standard deviation. The skewness statistics are large for all metrics in all applications except for the DIT metric. For the DIT metric, negative values for the peakedness statistic (kurtosis) for some systems indicate that the distribution of the DIT metric is flat. In Figure 1, the DIT is the closest to the normal distribution and there is no fat-tail behavior

Table 3: Descriptive statistics for JEdit system.

	CBO	NOC	DIT	RFC	WMC	NOM	NOV	SLOC
Mean	4.0	0.4	1.5	9.2	16.8	0.81	0.7	101.1
Median	2	0	1	3	5	0	0	30
Mode	1	0	1	1	1	0	0	5
Standard dev.	5.8	1.9	0.7	22.1	70.6	5.5	3.0	272.5
Kurtosis	49.9	164.6	0.2	109.7	471.6	323.4	104	156.1
Skewness	5.3	11.2	0.5	9.2	19.1	15.9	8.6	10.4
Minimum	0	0	0	0	0	0	0	1
Maximum	88	38	4	351	1936	134	53	5317

Table 4: Descriptive statistics for Openbravo ERP system.

	CBO	NOC	DIT	RFC	WMC	NOM	NOV	SLOC
Mean	5.4	0.7	2.5	39.6	24.2	0.62	1.2	135.6
Median	5	0	2	21	12	0	1	71
Mode	0	0	4	83	4	0	1	15
Standard dev.	4.8	10.9	1.4	36.2	42.7	3.7	2.9	221.7
Kurtosis	1.0	848.1	−1.6	−1.7	62.7	295.9	312.1	72.0
Skewness	0.9	28.7	0.0	0.4	6.6	.916	15.1	6.7
Minimum	0	0	0	0	0	0	0	0
Maximum	28	323	5	110	571	76	68	3425

Table 5: Descriptive statistics for Eclipse 3.0 system.

	CBO	NOC	DIT	RFC	WMC	NOM	NOV	SLOC
Mean	6.9	1.5	1.5	36.4	23.7	9.8	2.96	182.8
Median	4.0	.00	1.00	13.0	10	5.0	1.00	77.
Mode	0	0	1	0	2	1	0	17
Standard dev.	9.6	8.3	1.2	74.8	47	20.5	5.5	342.9
Kurtosis	4.0	16.5	1.6	6.4	7.2	17.7	9.1	6.5
Skewness	29.6	355.5	3.1	64.3	88.7	566.6	178.7	71.7
Minimum	0	0	0	0	0	0	0	2
Maximum	173	263	8	1357	1147	845	174	7719

Table 6: Descriptive statistics for JFreeChart 1.0.11 release.

	CBO	NOC	DIT	RFC	WMC	NOM	NOV	SLOC
Mean	4.2	0.6	1.9	33.8	21.2	0.7	0.8	122.5
Median	2	0	2	8	9	0	0	74
Mode	1	0	2	5	1	0	0	3
Standard dev.	6.2	2.0	0.97	68.6	39.1	2.3	2.3	185.7
Kurtosis	36.5	46.3	1.9	7.6	76.1	210.3	61.1	45.9
Skewness	4.2	5.9	0.8	3.0	7.1	13.0	6.8	5.4
Minimum	0	0	0	0	0	0	0	2
Maximum	87	25	6	320	582	44	30	2464

observed for the DIT metric in jBPM system. Therefore, the DIT metric is excluded from subsequent analysis of the power law. For other metrics, most classes are small in size and only few classes are large in complexity. The metrics that show skewness are left bounded (the minimum value is zero for all metrics) and usually unbounded from the right (maximum possible values are undefined) [42].

We study the evolution of JFreeChart. The JFreeChart size grows constantly as shown in Figure 2. The descriptive statistics for JFreeChart (version 1.0.11) are shown in Table 6, and they show the same behavior as other systems; that is, all metrics except DIT are skewed to the right. The maximum values are much larger than the mean, median, and standard deviation values for each metric. Therefore, we exclude

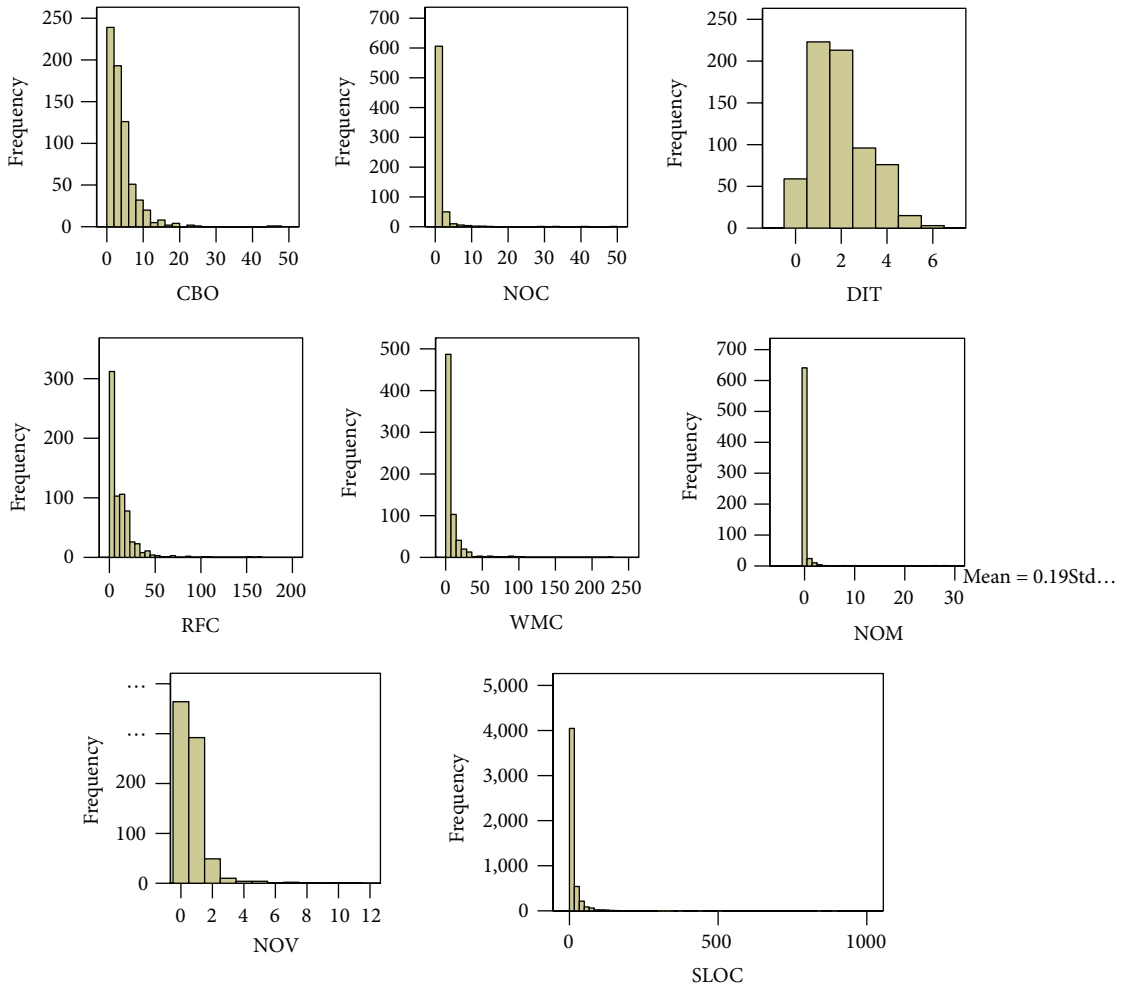

Figure 1: Histograms for jBPM metrics.

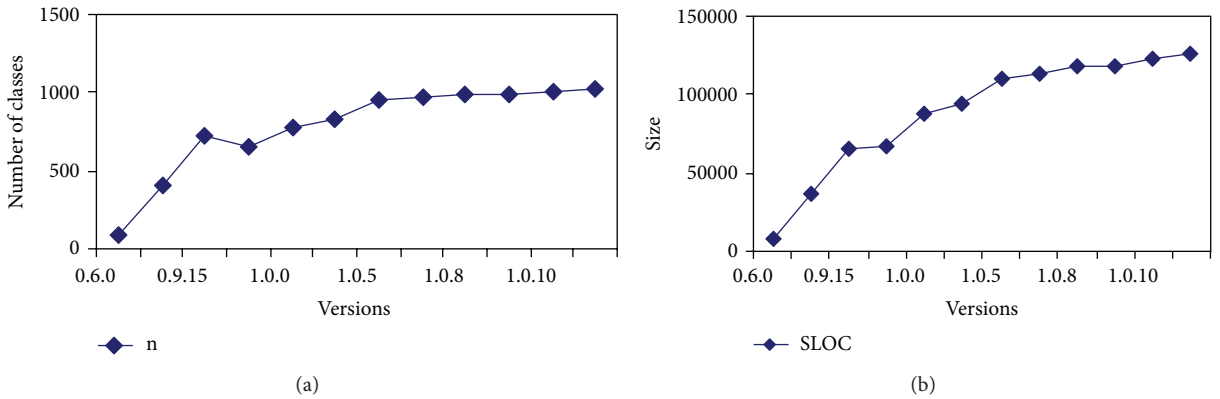

Figure 2: The evolution of JFreeChart size throughout 12 versions.

DIT from our analysis of the power law for JFreeChart as well.

4.2. Power Law Fitting. In this section, the fits of the power law to the seven metrics for all systems are conducted. Fitting to a power law using the maximum likelihood estimation produces the parameters (α, $x_{\min}$, and P value).

4.2.1. Fitting Power Law to the CBO Metric. CBO measures the coupling between objects. Coupling between objects

Table 7: Power law fit for CBO metric.

	α	$x_{\min}$	P	Mean	Std. dev.
jBPM	2.8	4	0.01	Finite	Infinite
jEdit	3.5	17	**0.85**	Finite	Finite
Openbravo ERP	3.5	8	0.00	Finite	Finite
Eclipse 3.0	3.5	25	**0.46**	Finite	Finite

reduces modularity and leaves classes interconnected. One of the well-known problems that high coupling might introduce is the God Class problem. A God Class provides services for too many other classes in the system. Such classes are usually very large in size as well. High coupling between objects was related to high maintenance effort. In addition, coupling is a complexity that is not easy to manage. Finding the most coupling is very important to manage software complexity. Consequently, the distribution of coupling is very important to understand and interpret the complexity of systems. Table 7 shows the power law fit for the CBO metric. Table 7 shows the estimated exponent, the minimum value, and the significance test. Column 5 shows whether the mean is finite or infinite based on the interpretation of the exponents. The statistical test shows that the power law could not be ruled out for only jEdit and Eclipse, that is, $P > 0.1$, whereas CBO cannot follow a power law for two systems, jBPM and Openbravo ERP. The exponents for jEdit and Eclipse are larger than 3, which indicate that the standard deviations for CBO metric in these systems are finite as shown in the last column. Figure 3 also demonstrates these results (weak fit to the line). CBO has a good fit to the line in two systems only. The graphical fit shows a power law behavior for CBO in only two systems, which is consistent with the statistical test.

The fit of the CBO metric for multiple releases does not show persistent pattern throughout all releases as shown in Table 8. CBO fits to the power law are ruled out for nine versions, excluding 3 versions that show possibility to follow a power law distribution. The estimates of the MLE $(\alpha, x_{\min})$ are not persistent for multiple versions of the JFreeChart; that is, they range between 1.87 and 3.5. Exponents that are less than 2 mean that the data is not strongly skewed to the right, while values that are larger than 3 mean that few classes have the most coupling. The CBO metric does not show an evidence of a power law behavior in all releases.

4.2.2. Fitting Power Law for the NOC Metric. The NOC metric measures the amount of abstraction of a class. The number of children for a module is an indicator of the amount of reuse. In Table 9, we observe the power law distribution for the NOC metric. The power law could not be ruled out for all systems, that is, $P > 0.1$. Figure 4 also demonstrates a fit to line which confirms a power law behavior. The exponents for all projects are less than 3, hence the standard deviation for these metrics is infinite. From the statistical tests and the graphical fits, we can conclude that NOC may follow a power law distribution.

The fit of the NOC metric for multiple releases is shown in Table 10. The distribution of the NOC metric could not be ruled out to follow a power law for most versions. The results

Table 8: Power law fit for the CBO metric in JFreeChart releases.

Version	α	$x_{\min}$	P	Mean	Std. dev.
0.6.0	2.28	3	0.07	Finite	Infinite
0.9.4	3.5	9	**0.25**	Finite	Finite
0.9.15	3.5	21	**0.78**	Finite	Finite
0.9.21	1.87	2	0.00	Infinite	Infinite
1.0.0	3.5	12	0.01	Finite	Finite
1.0.2	3.5	12	0.01	Finite	Finite
1.0.5	1.86	2	0.00	Infinite	Infinite
1.0.6	1.85	2	0.00	Infinite	Infinite
1.0.8	1.87	2	0.00	Infinite	Infinite
1.0.9	1.87	2	0.00	Infinite	Infinite
1.0.10	1.87	2	0.00	Infinite	Infinite
1.0.11	3.5	15	**0.41**	Finite	Finite

Table 9: Power law fit for the NOC metric.

	α	$x_{\min}$	P	Mean	Std. dev.
jBPM	2.2	2	**0.68**	Finite	Infinite
jEdit	2.32	2	**0.37**	Finite	Infinite
Openbravo ERP	1.66	1	**0.10**	Infinite	Infinite
Eclipse	2.33	13	**0.53**	Finite	Infinite

Table 10: Power law fit for the NOC metric in JFreeChart releases.

Version	α	$x_{\min}$	P	Mean	Std. dev.
0.6.0	3.25	2	**0.40**	Finite	Infinite
0.9.4	2.72	3	**0.66**	Finite	Infinite
0.9.15	2.26	2	0.05	Finite	Infinite
0.9.21	2.25	2	0.06	Finite	Infinite
1.0.0	3.34	6	**0.62**	Finite	Finite
1.0.2	2.17	2	0.04	Finite	Infinite
1.0.5	3.07	5	**0.35**	Finite	Finite
1.0.6	3.42	6	**0.43**	Finite	Finite
1.0.8	3.44	6	**0.62**	Finite	Finite
1.0.9	3.4	6	**0.80**	Finite	Finite
1.0.10	3.31	6	**0.81**	Finite	Finite
1.0.11	3.23	6	**0.93**	Finite	Finite

are persistent for the last six versions. For these versions, the exponent changes slightly. Therefore, for multiple releases and most systems under study, the power law fit could not be ruled out for the NOC metric. Since the exponents are larger than three for all persistent releases, the expected mean and expected standard deviation are finite and bounded. Therefore, reuse may follow a power law behavior where few classes have the most reuse, while most classes are rarely reused.

4.2.3. Fitting Power Law for the RFC Metric. The FRC metric measures the amount of responsibilities of a class. The equal distribution of responsibility is an indicator of a good design. Otherwise, God classes may appear in the system, which are difficult to maintain and test. Table 11 shows the observations of a power law fit for the RFC metric. The power law could

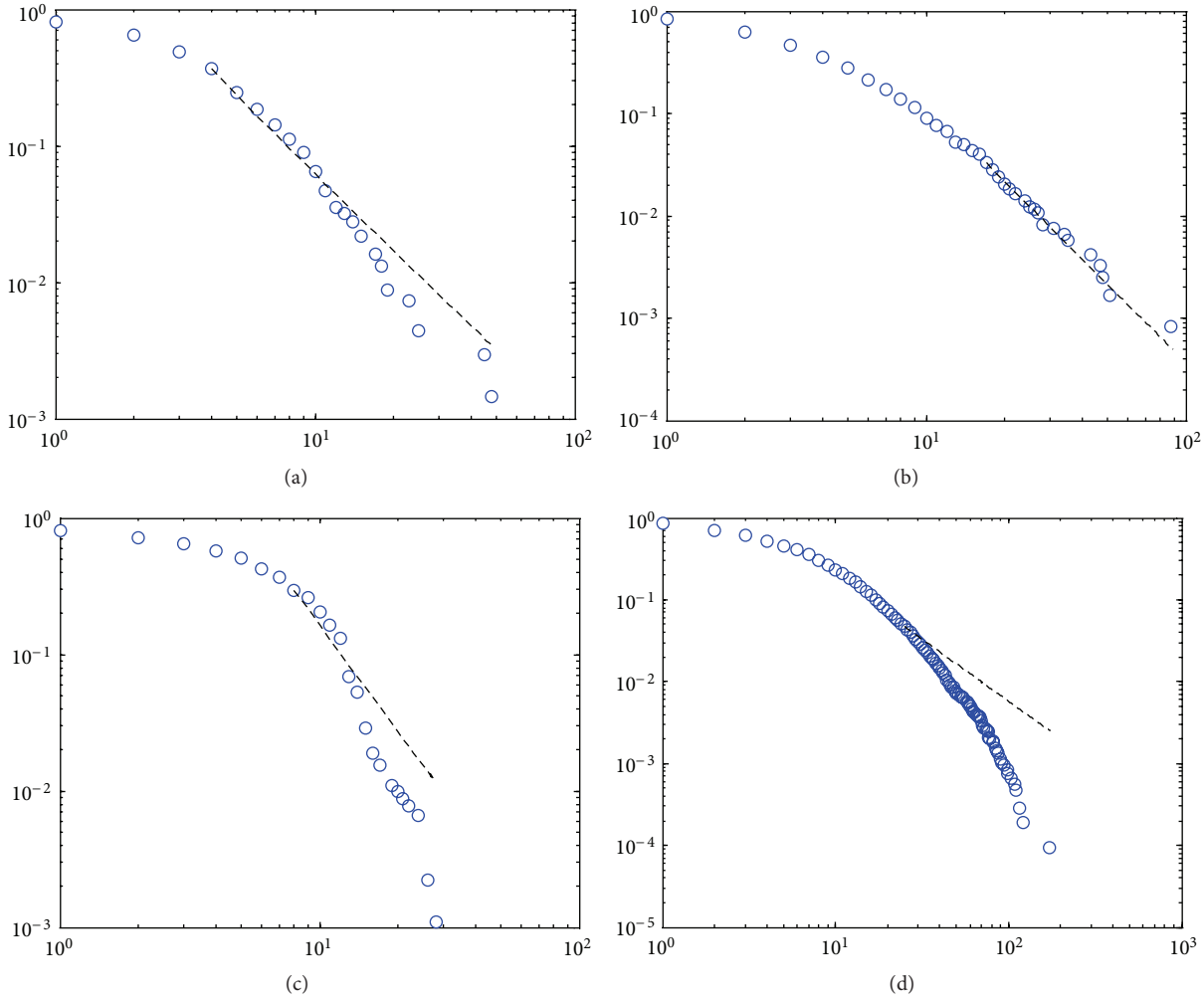

Figure 3: The cumulative distribution functions $P(x)$ and their maximum likelihood power law fits for the CBO metric (jBPM, jEdit, Openbravo ERP, and Eclipse).

Table 11: Power law fit for the RFC metric.

	α	$x_{\min}$	P	Mean	Std. dev.
jBPM	3.09	17	**0.69**	Finite	Finite
jEdit	2.52	24	**0.76**	Finite	Infinite
Openbravo ERP	1.51	5	0.00	Infinite	Infinite
Eclipse	2.32	48	0.00	Finite	Infinite

not be ruled out for two systems, that is, $P > 0.1$. Figure 5 also demonstrates these fits. The graphical fits for RFC in OpenBravo and Eclipse are clearly demonstrated not to follow a power law. The exponent for JEdit is less than 3, which means that the standard deviation is infinite.

The fits of the RFC metric in Table 12 do not show persistent pattern throughout all releases. RFC fits to the power law are ruled out for 11 versions, excluding the first one that shows possibility to follow a power law distribution. The estimates of MLE (α, $x_{\min}$) are not persistent for multiple versions. Therefore, we can conclude that the RFC metric does not follow a power law distribution.

Table 12: Power law fit for the RFC metric in JFreeChart releases.

Version	α	$x_{\min}$	P	Mean	Std. dev.
0.6.0	3.5	42	**0.24**	Finite	Finite
0.9.4	2.48	32	0.00	Finite	Infinite
0.9.15	1.66	3	0.00	Infinite	Infinite
0.9.21	1.66	4	0.00	Infinite	Infinite
1.0.0	1.78	5	0.00	Infinite	Infinite
1.0.2	1.77	6	0.00	Infinite	Infinite
1.0.5	1.79	5	0.00	Infinite	Infinite
1.0.6	1.78	5	0.00	Infinite	Infinite
1.0.8	1.77	5	0.00	Infinite	Infinite
1.0.9	1.77	5	0.00	Infinite	Infinite
1.0.10	1.76	5	0.00	Infinite	Infinite
1.0.11	1.75	5	0.00	Infinite	Infinite

4.2.4. Fitting Power Law for the WMC Metric. The WMC measures the amount of complexity of a class by aggregating the complexity of all methods in a class. A large value of

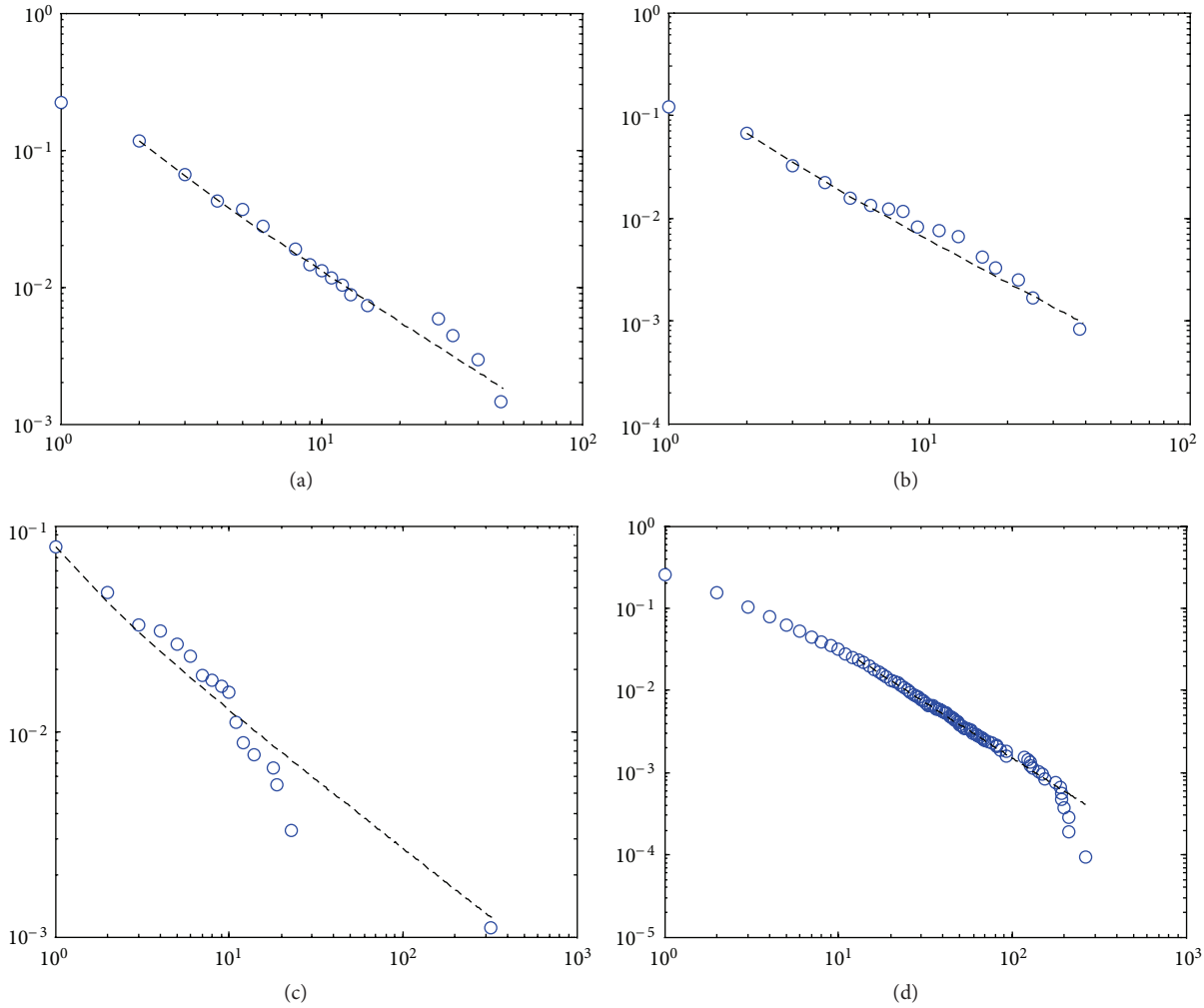

Figure 4: The cumulative distribution functions $P(x)$ and their maximum likelihood power law fits for the NOC metric (jBPM, jEdit, Openbravo ERP, and Eclipse).

this metric is an indicator of a low quality. The distribution of complexity among classes can be used by developers to take action to improve software quality. Table 13 shows the observations of a power law fit for the WMC metric. The power law could not be ruled out for two systems, that is, $P > 0.1$. Figure 6 also demonstrates a graphical fit to the line. The exponents are less than 3, which means that the standard deviations for these metrics are infinite.

The fits of the WMC metric for multiple releases are shown in Table 14. The distribution of the WMC metric could not be ruled out to follow a power law for most versions (10 out of 12) as shown in Table 14. The results are persistent for the last seven versions. For these versions, the exponents change from 2.5 insignificantly. This behavior indicates a scale-free behavior for the WMC metric. Additionally, the results of WMC fits to other systems, shown in Table 14, also show that the WMC metric could not be ruled out for two systems only. Therefore, the power law fit for multiple releases and most systems under study could not be ruled out for the WMC metric. Furthermore, we can conclude that WMC does not always follow a power law. Since the exponent values are larger than two for all releases, the expected mean values are finite and bounded, while the standard deviations are infinite because the exponents are less than three.

Table 13: Power law fit for the WMC metric.

	α	$x_{\min}$	P	Mean	Std. dev.
jBPM	2.69	18	0.08	Finite	Infinite
jEdit	2.13	17	**0.24**	Finite	Infinite
Openbravo ERP	2.91	46	**0.32**	Finite	Infinite
Eclipse	2.49	43	0.00	Finite	Infinite

4.2.5. Fitting Power Law for the SLOC Metric. SLOC is a well-known size measure. SLOC can be used as an indicator of software development effort. The distribution of SLOC shows the large classes. Table 15 shows the observation of a power law fit for the SLOC metric. The power law could not be ruled out for two systems, that is, $P > 0.1$. Figure 7 also demonstrates these fits; the deviations from the line are clearly demonstrated for jBPM and Eclipse. The exponent for

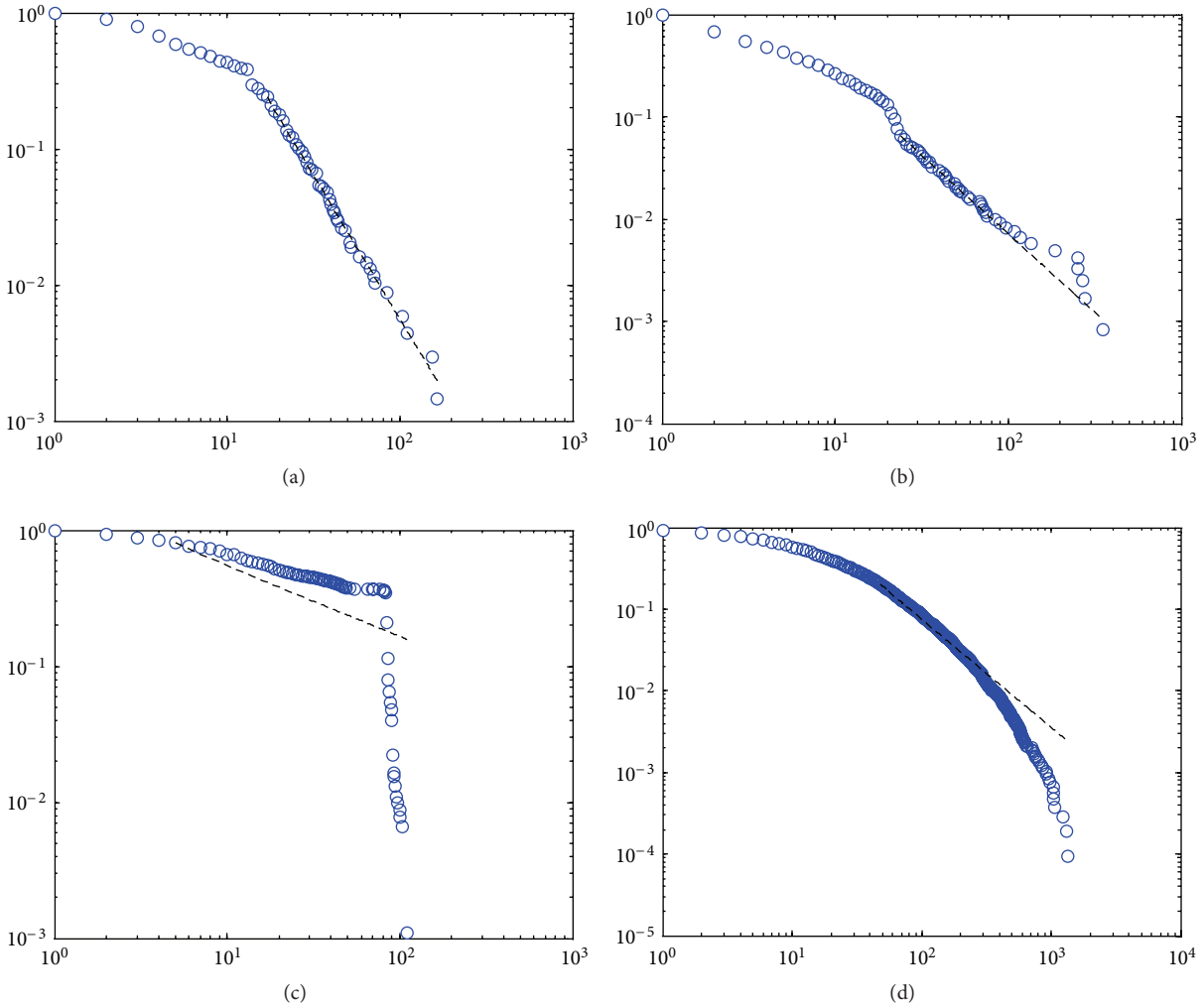

Figure 5: The cumulative distribution functions $P(x)$ and their maximum likelihood power law fits for the RFC metric (jBPM, jEdit, Openbravo ERP, and Eclipse).

Table 14: Power law fit for the WMC metric in JFreeChart releases.

Version	α	$x_{\min}$	P	Mean	Std. dev.
0.6.0	2.16	29	0.02	Finite	Infinite
0.9.4	2.96	27	**0.62**	Finite	Infinite
0.9.15	2.6	33	**0.29**	Finite	Infinite
0.9.21	2.9	56	**0.57**	Finite	Infinite
1.0.0	2.47	29	0.03	Finite	Infinite
1.0.2	2.48	29	**0.12**	Finite	Infinite
1.0.5	2.58	29	**0.22**	Finite	Infinite
1.0.6	2.56	29	**0.17**	Finite	Infinite
1.0.8	2.51	27	**0.15**	Finite	Infinite
1.0.9	2.51	27	**0.37**	Finite	Infinite
1.0.10	2.51	26	**0.62**	Finite	Infinite
1.0.11	2.57	29	**0.65**	Finite	Infinite

jEdit is less than 3, which means that the standard deviation is infinite.

Table 15: Power law fit for the SLOC metric.

	α	$x_{\min}$	P	Mean	Std. dev.
jBPM	2.29	24	0.06	Finite	Infinite
jEdit	2.54	235	**0.37**	Finite	Infinite
Openbravo ERP	3.09	279	**0.56**	Finite	Finite
Eclipse	2.81	543	0.02	Finite	Infinite

The fit of the power law for multiple releases shows that the SLOC metric could not be ruled out for most versions (10 out of 12) as shown in Table 16. The results are persistent for the last nine versions only. Additionally, the results of SLOC fits that were shown in Table 15 also show that the SLOC metric could not be ruled out for two systems only. Since the exponent values are larger than two for all releases, the expected mean values are finite and bounded, while the standard deviations are infinite because the exponents are less than three.

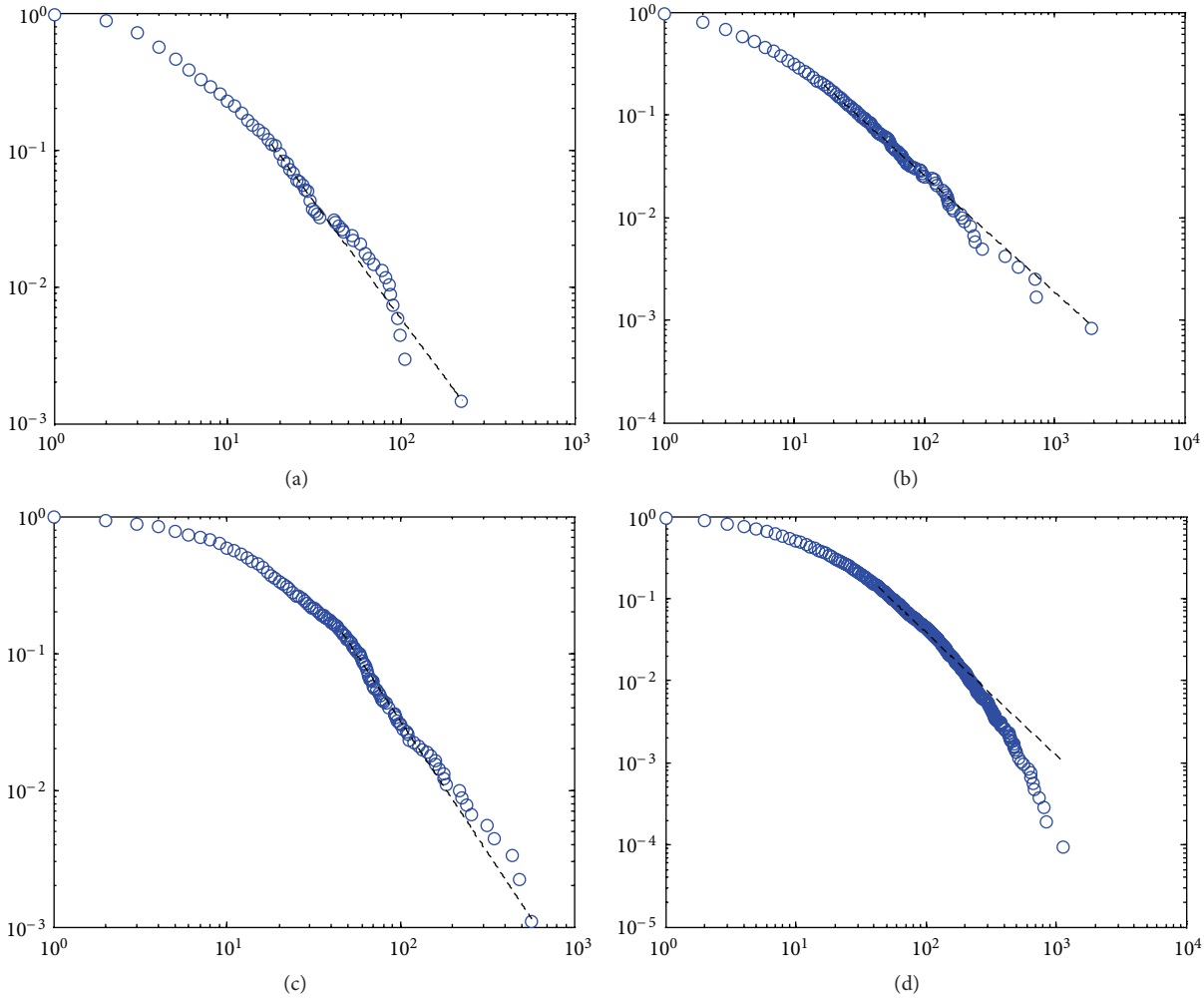

Figure 6: The cumulative distribution functions $P(x)$ and their maximum likelihood power law fits for the WMC metric (jBPM, jEdit, Openbravo ERP, and Eclipse).

Table 16: Power law fit for the SLOC metric in JFreeChart releases.

Version	α	$x_{\min}$	P	Mean	Std. dev.
0.6.0	2.66	106	0.02	Finite	Infinite
0.9.4	3.24	200	**0.80**	Finite	Finite
0.9.15	2.05	35	0.00	Finite	Infinite
0.9.21	2.62	173	**0.16**	Finite	Infinite
1.0.0	2.27	71	**0.12**	Finite	Infinite
1.0.2	2.36	90	**0.11**	Finite	Infinite
1.0.5	2.61	138	**0.36**	Finite	Infinite
1.0.6	2.58	139	**0.30**	Finite	Infinite
1.0.8	2.62	145	**0.29**	Finite	Infinite
1.0.9	2.63	145	**0.44**	Finite	Infinite
1.0.10	2.65	149	**0.24**	Finite	Infinite
1.0.11	2.63	148	**0.41**	Finite	Infinite

4.2.6. Fitting Power Law for the NOM Metric. NOM measures the interface of a class and provides the services of a class. Table 17 shows the observations of a power law fit for the NOM metric. The results show that the NOM could not be ruled out for three systems. Figure 8 also demonstrates these fits. The exponents for all projects are less than 3, whereas the standard deviations for these metrics are infinite.

Table 17: Power law fit for the NOM metric.

	α	$x_{\min}$	P	Mean	Std. dev.
jbpm	2.02	1	**0.40**	Finite	Infinite
jedit	2.39	11	**0.64**	Finite	Infinite
Openbravo ERP	2.36	3	**0.40**	Finite	Infinite
Eclipse	2.96	32	0.02	Finite	Infinite

The fit of the power law for multiple releases shows that the NOM metric could not be ruled out for most versions (9 out of 12) as shown in Table 18. The results are persistent for the last nine versions only. Additionally, the results of SLOC fits that were shown in Table 17 also show that the NOM metric could not be ruled out for three systems. Since the exponent values are larger than two for all releases,

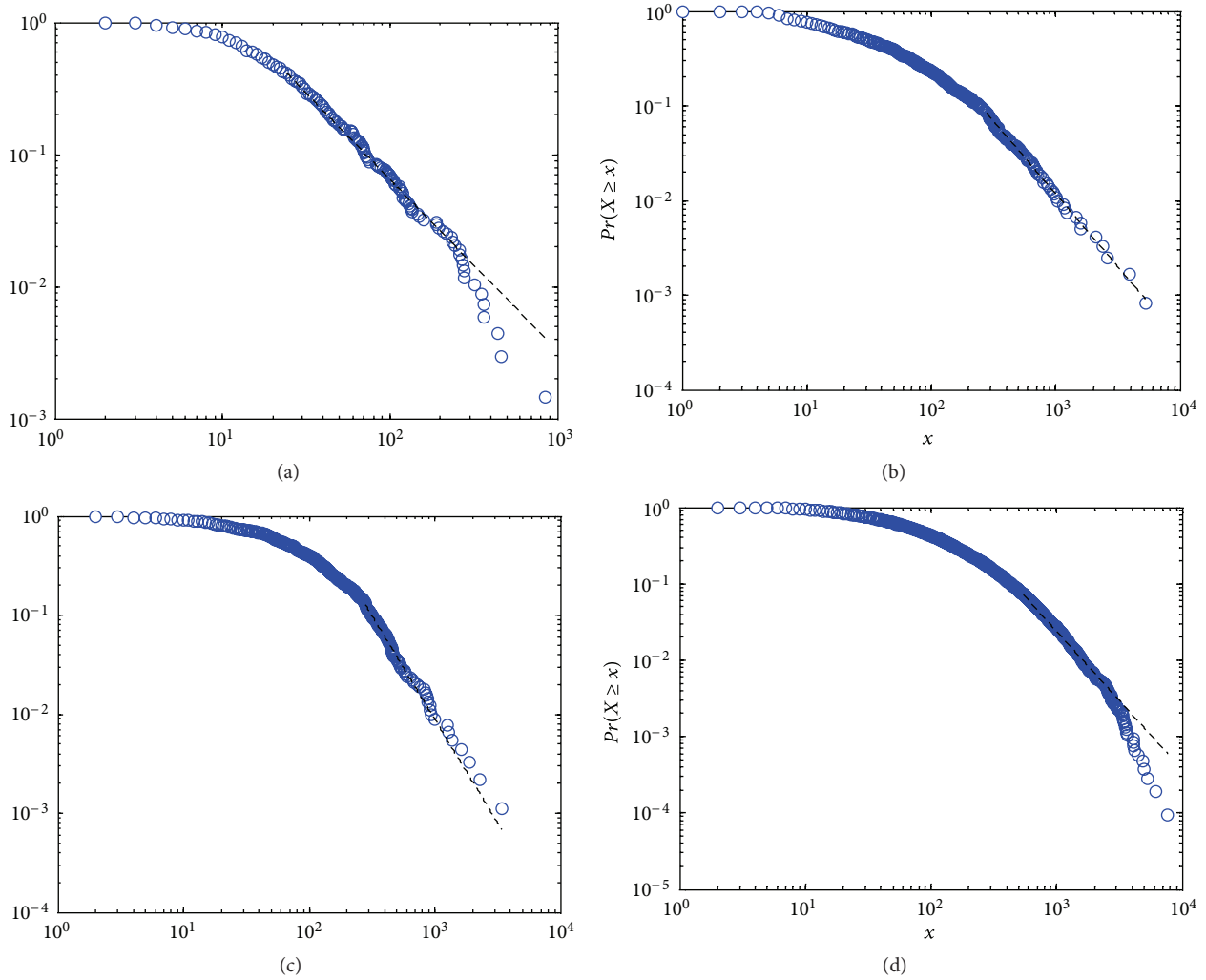

Figure 7: The cumulative distribution functions $P(x)$ and their maximum likelihood power law fits for the SLOC metric (jBPM, jEdit, Openbravo ERP, and Eclipse).

Table 18: Power law fit for the NOM metric in JFreeChart releases.

Version	α	$x_{\min}$	P	Mean	Std. dev.
0.6.0	2.07	1	0.001	Finite	Infinite
0.9.4	2.4	1	0.019	Finite	Infinite
0.9.15	2.15	2	**0.477**	Finite	Infinite
0.9.21	2.87	1	0.001	Finite	Infinite
1.0.0	2.67	1	**0.155**	Finite	Infinite
1.0.2	2.6	1	**0.0128**	Finite	Infinite
1.0.5	2.71	1	**0.407**	Finite	Infinite
1.0.6	2.72	1	**0.359**	Finite	Infinite
1.0.8	2.73	1	**0.47**	Finite	Infinite
1.0.9	2.72	1	**0.393**	Finite	Infinite
1.0.10	2.71	1	**0.319**	Finite	Infinite
1.0.11	2.72	1	**0.512**	Finite	Infinite

the expected mean values are finite and bounded, while the standard deviations are infinite because the exponents are less than three.

Table 19: Power law fit for the NOV metric.

	α	$x_{\min}$	P	Mean	Std. dev.
jBPM	2.87	1	**0.30**	Finite	Infinite
jEdit	2.49	6	**0.24**	Finite	Infinite
Openbravo ERP	2.62	1	**0.93**	Finite	Infinite
Eclipse	3.5	15	**0.41**	Finite	Finite

4.2.7. Fitting Power Law for the NOV Metric. NOV measures the data in a class. Table 19 shows the observations of a power law fit for the NOV metric. The results show that the NOV could not be ruled out for all systems. Figure 9 also demonstrates these fits. The exponents for three projects are less than 3, which means that the standard deviations for these metrics are infinite.

The distribution of the NOV metric could not be ruled out to follow a power law for all versions as shown in Table 20. The results are persistent for the last eight versions. For these

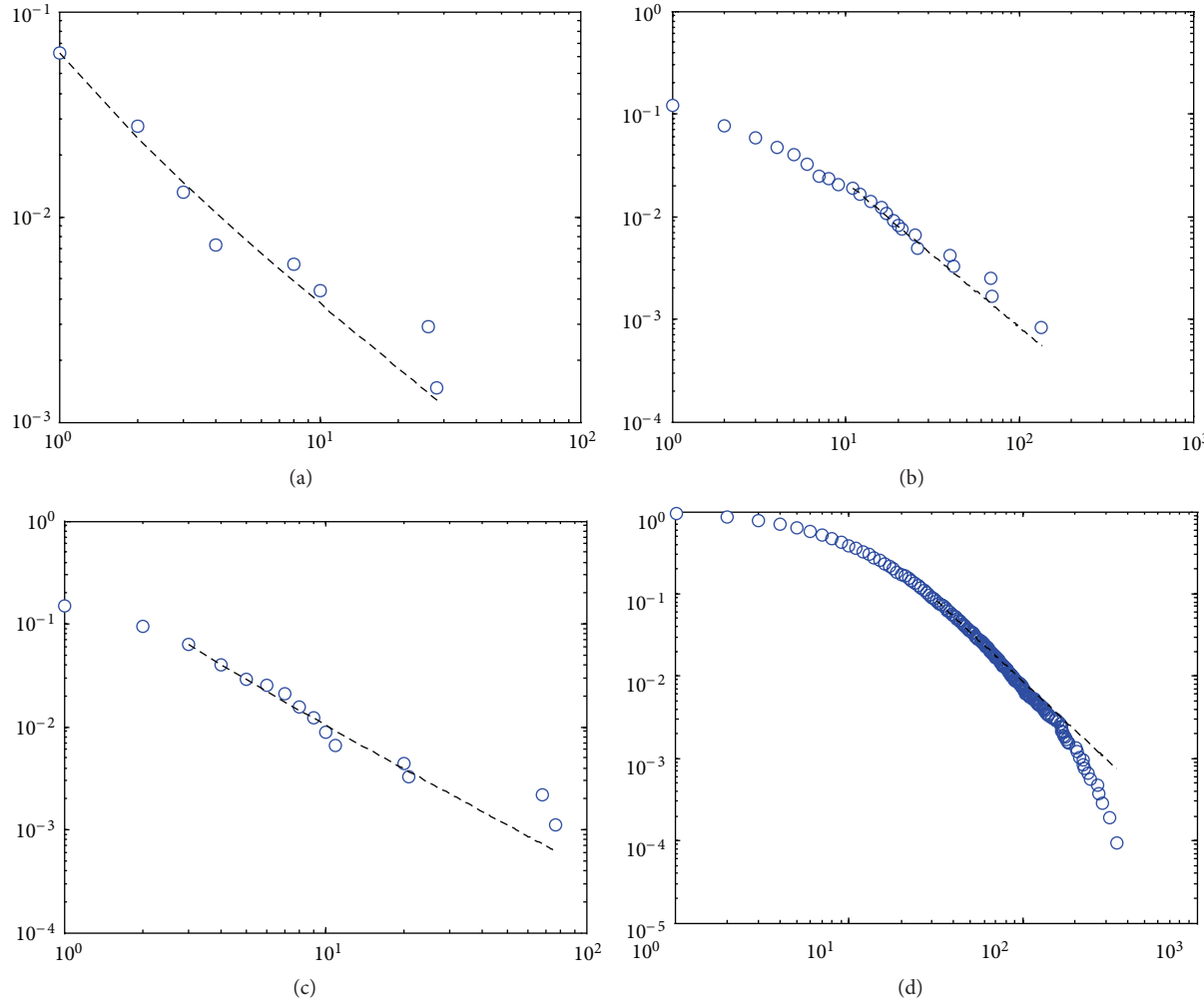

Figure 8: The cumulative distribution functions $P(x)$ and their maximum likelihood power law fits for the NOM metric (jBPM, jEdit, Openbravo ERP, and Eclipse).

Table 20: Power law fits for the NOV metric in JFreeChart releases.

Version	α	$x_{\min}$	P	Mean	Std. dev.
0.6.0	3.5	6	**0.63**	Finite	Finite
0.9.4	2.01	2	**0.31**	Finite	Infinite
0.9.15	2.31	3	**0.30**	Finite	Infinite
0.9.21	2.41	3	**0.79**	Finite	Infinite
1.0.0	2.73	4	**0.27**	Finite	Infinite
1.0.2	2.69	4	**0.46**	Finite	Infinite
1.0.5	2.67	4	**0.29**	Finite	Infinite
1.0.6	2.71	4	**0.52**	Finite	Infinite
1.0.8	2.69	4	**0.59**	Finite	Infinite
1.0.9	2.69	4	**0.56**	Finite	Infinite
1.0.10	2.68	4	**0.65**	Finite	Infinite
1.0.11	2.71	4	**0.62**	Finite	Infinite

versions the exponent changes from 2.7 insignificantly. Additionally, the results of NOV fits that were shown in Table 19 also show that the NOV metric could not be ruled out for all systems. Therefore, for multiple releases and most systems under study, the power law fit could not be ruled out for the NOV metric, and NOV follows a power law as shown from the data analysis. Since the exponent values are larger than two for all releases, the expected mean values are finite and bounded, while the standard deviations are infinite because the exponent values are less than three.

5. Power Law Applications

A metric that follows a power law means that the data can be divided into two groups, $< x_{\min}$ and $\geq x_{\min}$. The data above the $x_{\min}$ has the characteristics of scale-free networks, where small number of classes is responsible for the largest complexity in software systems. Setting threshold values can be one of the applications that utilize the existence of a power law behavior. Thresholds are breakpoints that separate classes into different groups; each group has a distinct behavior (complexity). The application of threshold values helps developers and testers in highlighting the most complex

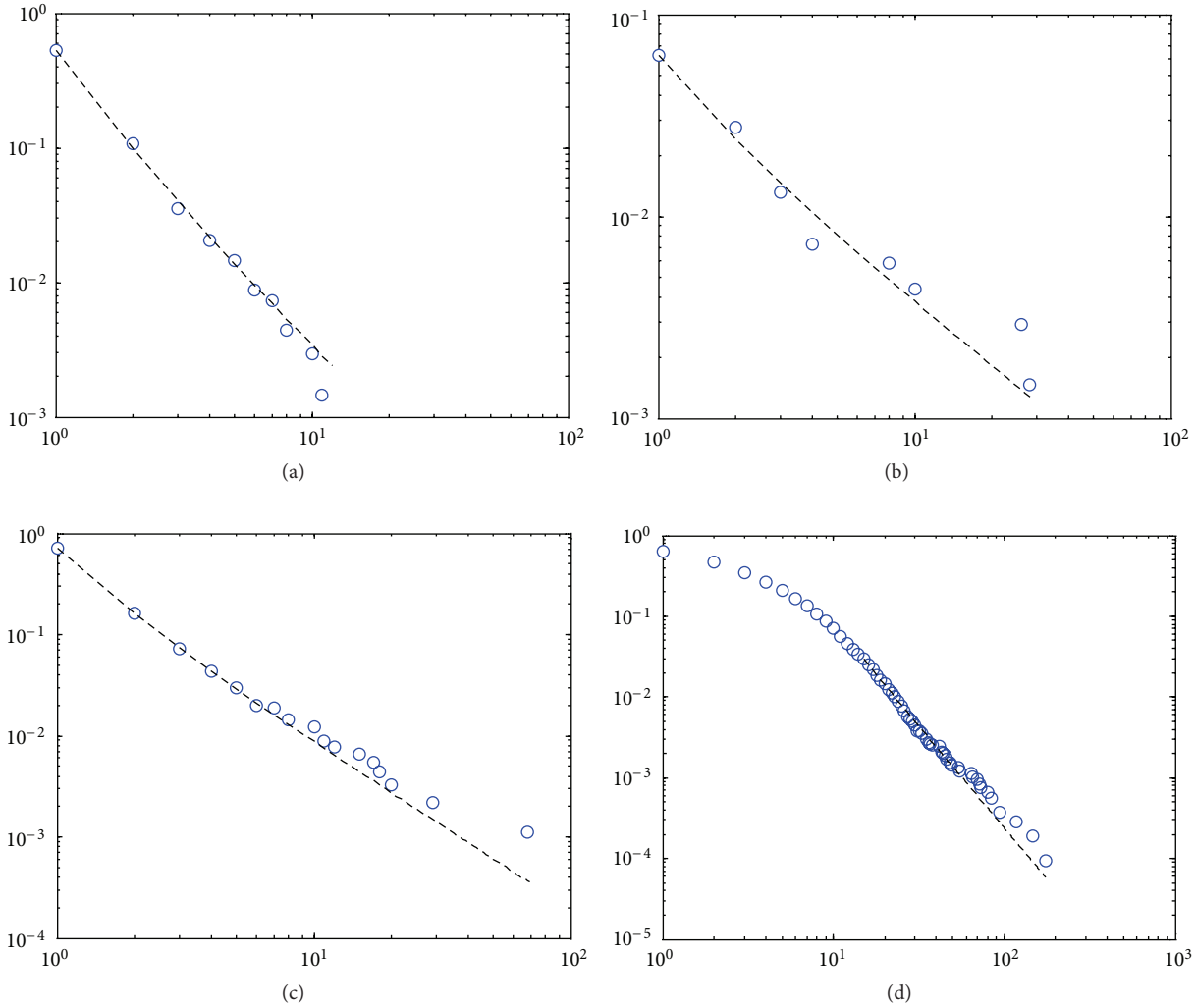

Figure 9: The cumulative distribution functions $P(x)$ and their maximum likelihood power law fits for the NOV metric (jBPM, jEdit, Openbravo ERP, and Eclipse (to be added)).

classes in the system. For example, threshold values were used in many studies to identify the potential for refactoring. Gronback used thresholds to identify six refactoring opportunities (Refused Bequest, God Class, God Methods, Shotgun Surgery, Feature Envy, and Data Class) [14]. Threshold values for complexity metrics are expected to appear in the second part of the data that follows a power law behavior. Therefore, these threshold values should be larger than or equal to the $x_{\min}$ for the metrics under investigation. Threshold values were identified using many techniques in previous research. Rosenberg suggested a set of threshold values for the CK metrics that can be used to select classes for inspection or redesign [16]. Shatnawi also identified threshold values based on a logistic regression of the relationship between software metrics and faults in a large open-source system (Eclipse IDE) [43]. In another study, Shatnawi et al. also reported threshold values based on the ROC analysis [44]. Previous researches also have shown that there are many threshold values that were identified for a particular metric; for example, the CBO metric has three different thresholds. Lanza and Marinescu noted that threshold values can be defined based on reasonable arguments [45]. Characteristics of a power law distribution can help in proposing criteria for reasonable threshold values.

The power law distribution is strongly connected with other distributions such as the Pareto distribution (often known as the 20 : 80 rule) [46]. The Pareto law is named after the economist Vilfredo Pareto, who proposed a statistical model to describe the distribution of wealth among individuals [47]. The distribution is expressed simply as the "20 : 80" rule, which states that 20% of the population has 80% of the wealth. Threshold values can be relative such as the 20 : 80 rule; that is, 20% of classes have 80% of the complexity. The exact relationship may not be 20 : 80. However, 20 : 80 can be used as metaphor or a useful hypothesis, and it is not always expected. Practically, this necessitates that, after a certain threshold, the marginal cost of improving a situation further becomes prohibitively high. In other words, the 20 : 80 rule means that in any data sample a few (20%) are vital and many (80%) are considered trivial. The Pareto law can tell where

the majority of the distribution of a variable lies. Newman [22] has provided the relationship between the fraction of wealth (W) and the population (P), which is shown in [22]

$$W = P^{(\alpha-2)/(\alpha-1)}. \quad (10)$$

Given that $\alpha > 2$, the Pareto law can be used to find where the majority of the classes that have the most complexity exist. That is, a large amount of complexity is concentrated in a small number of classes. Equation (10) can be used to plan the resources of software development for the vital parts of the system. Figure 10 shows the Pareto graph for metrics in jEdit (a measure of data in modules). The fraction W can be used to set threshold values. For example, according to Figure 10, 60% of the data exists in only 20% of modules. This graph can be used to select the appropriate threshold values. For example, in SLOC, we notice a 20 : 70 rule; that is, 20% of classes have 70% of the code. We notice 20 : 83 rule for the WMC which indicates that most complexity appears in 20% of classes. Therefore, if the investigation targets problems caused by the data in a system, then the heavy tail includes a small number of vital modules. Therefore, the investigation is limited to a small part of the system instead of the entire system.

Threshold values can be absolute values that divide the data into two groups. In the following, the $x_{\min}$ is proposed as a reasonable threshold. The effects of the proposed thresholds are tested on an open-source system JEdit. All metrics of the JEdit have a potential to follow a power law as shown in the previous sections. The fault data of the JEdit were reported online and publicly available (http://promisedata.org/). We then use a data mining technique, Random Forest Trees, to find the relationships between software metrics and faults. We have built three prediction models: (1) for the classes that are larger than or equal to the threshold value, (2) for classes that are less than the threshold value, and (3) for all classes. We use Weka (http://www.cs.waikato.ac.nz/ml/weka/) to run a 10-fold classification on the three data sets. To evaluate the differences between the three classifications for each metric, we have calculated the Precision and the Recall. We use the F-measure which combines both Precision and Recall as a summary metric. We refer the readers to the Appendix B and [48, 49] for more information on the Random Forrest Trees and the evaluation metrics. The results of the Radom Forests classifier on the three data sets are shown in Table 21. For all metrics except NOC, the classes that are larger than the threshold have a reasonable classification performance, while for the classes less than the threshold, the F-measure values are smaller. The NOC metrics, however, have opposite direction; that is, F-measure is high for the classes less than the threshold. Therefore, the threshold values have significant effects in improving the classification of fault-prone classes. These results show that power law certainly has characteristics that can be used to understand the nature of threshold values. This behaviour is consistent with the second law of the software evolution; that is, "as a program is evolved its complexity increases unless work is done to maintain or reduce it" [50].

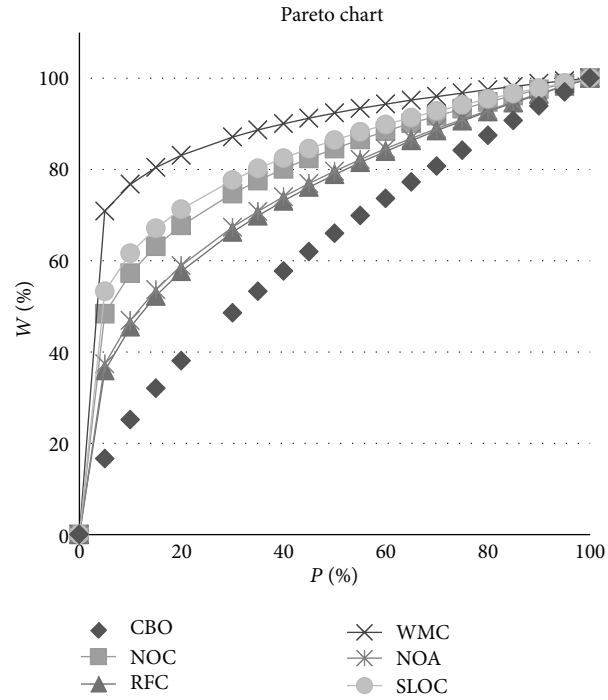

Figure 10: The fraction W of the total data in jEdit.

Table 21: Classification performance for the three data sets.

	Precision	Recall	F-measure*
CBO ≥ 17	70%	83%	**76**%
CBO < 17	51%	50%	50%
CBO	58%	60%	59%
NOC ≥ 2	8%	7%	7%
NOC < 2	57%	100%	**72**%
NOC	56%	94%	70%
RFC ≥ 24	66%	66%	**66**%
RFC < 24	50%	35%	41%
RFC	62%	58%	60%
WMC ≥ 17	81%	88%	**84**%
WMC < 17	56%	48%	52%
WMC	63%	51%	56%
LOC ≥ 235	78%	82%	**80**%
LOC < 235	52%	50%	51%
LOC	62%	64%	63%
NOM ≥ 11	78%	85%	**81**%
NOM < 11	62%	66%	64%
NOM	62%	72%	67%

* The NOV is not available in the Promise data.

6. Conclusions

Software metrics are believed to follow a power law distribution. In this paper, a statistical assessment of the behavior of software systems is proposed. This statistical assessment is conducted on object-oriented metrics for five different systems. The systems under investigation are divided into

two contexts, four systems of different sizes and twelve releases resulting from the evolution of an open-source system. In summary, five metrics, NOC, WMC, NOM, NOV, and SLOC, have shown a potential to follow a power law distribution. Two metrics, RFC and CBO, do not follow a power law behavior. The power law characteristics can be used to set threshold values for software metrics or to find the maximum value of a particular metric. Exponent values are found not consistent across different projects; therefore, thresholds should be estimated for each project separately. For systems that follow a power law distribution, the central limit theorem does not hold. Rather, extreme value theory (i.e., tail-fitting approach) should be considered in evaluating threshold values. We applied the use of threshold values on fault prediction and found better fault predictions for classes above the threshold values. For the future, we plan to use power law characteristic to identify potential code refactorings.

Appendices

A. Systems under Study

jBPM is a workflow management system. Business processes are defined in an expressive language and deployed along custom resources inside a process archive. jBPM offers tasks to users, runs the automated actions, and maintains the state and audit log.

jEdit is a programmer's text editor written in Java. It uses the Swing toolkit for the GUI and can be configured as a rather powerful IDE through the use of its plugin architecture.

Openbravo ERP is the professional web-based open-source ERP solution providing unique high-impact benefits: (1) comprehensive, (2) innovative, and (3) cost effective.

Eclipse is a multilanguage software development environment comprising an integrated development environment (IDE) and an extensible plug-in system.

JFreeChart is a free (LGPL) chart library for the Java(tm) platform. It supports bar charts, pie charts, line charts, time series charts, scatter plots, histograms, simple Gantt charts, Pareto charts, bubble plots, dials, thermometers, and more.

B. Random Forest Trees

Random forests are extended form decision trees. Random forests build many classification trees (hundreds or even thousands) using subsets of the training data. To classify a module, use the software metrics as input for all trees in the forest to find a classification. The forest chooses the classification having received the most predictions (votes) from all trees.

References

[1] V. R. Basili, L. C. Briand, and W. L. Melo, "A validation of object-oriented design metrics as quality indicators," *IEEE Transactions on Software Engineering*, vol. 22, no. 10, pp. 751–761, 1996.

[2] T. Gyimóthy, R. Ferenc, and I. Siket, "Empirical validation of object-oriented metrics on open source software for fault prediction," *IEEE Transactions on Software Engineering*, vol. 31, no. 10, pp. 897–910, 2005.

[3] R. Shatnawi and W. Li, "The effectiveness of software metrics in identifying error-prone classes in post-release software evolution process," *The Journal of Systems and Software*, vol. 81, no. 11, pp. 1868–1882, 2008.

[4] S. R. Chidamber and C. F. Kemerer, "Metrics suite for object oriented design," *IEEE Transactions on Software Engineering*, vol. 20, no. 6, pp. 476–493, 1994.

[5] R. M. Szabo and T. M. Khoshgoftaar, "Assessment of software quality in a C++ environment," in *Proceedings of the 6th International Symposium on Software Reliability Engineering*, pp. 240–249, October 1995.

[6] L. C. Briand, J. W. Daly, and J. Wüst, "A unified framework for cohesion measurement in object-oriented systems," *Empirical Software Engineering*, vol. 3, no. 1, pp. 65–117, 1998.

[7] L. C. Briand and J. W. Daly, "A unified framework for coupling measurement in object-oriented systems," *IEEE Transactions on Software Engineering*, vol. 25, no. 1, pp. 91–121, 1999.

[8] L. C. Briand, J. Wüst, J. W. Daly, and D. V. Porter, "Exploring the relationships between design measures and software quality in object-oriented systems," *The Journal of Systems and Software*, vol. 51, no. 3, pp. 245–273, 2000.

[9] M. Cartwright and M. Shepperd, "An empirical investigation of an object-oriented software system," *IEEE Transactions on Software Engineering*, vol. 26, no. 8, pp. 786–796, 2000.

[10] N. E. Fenton and N. Ohlsson, "Quantitative analysis of faults and failures in a complex software system," *IEEE Transactions on Software Engineering*, vol. 26, no. 8, pp. 797–814, 2000.

[11] K. El Emam, S. Benlarbi, N. Goel, and S. N. Rai, "The confounding effect of class size on the validity of object-oriented metrics," *IEEE Transactions on Software Engineering*, vol. 27, no. 7, pp. 630–648, 2001.

[12] R. Subramanyam and M. S. Krishnan, "Empirical analysis of CK metrics for object-oriented design complexity: implications for software defects," *IEEE Transactions on Software Engineering*, vol. 29, no. 4, pp. 297–310, 2003.

[13] R. Marinescu, "Measurement and quality in object-oriented design," in *Proceedimgs of the 21st IEEE International Conference on Software Maintenance (ICSM '05)*, pp. 701–704, September 2005.

[14] R. C. Gronback, "Software remodeling: improving design and implementation quality, using audits, metrics and refactoring in Borland Together Control Center," A Borland White Paper, 2003.

[15] W. Li and R. Shatnawi, "An empirical study of the bad smells and class error probability in the post-release object-oriented system evolution," *The Journal of Systems and Software*, vol. 80, no. 7, pp. 1120–1128, 2007.

[16] L. Rosenberg, "Applying and interpreting object oriented metrics," Software Assurance Technology Center at NASA Goddard Space Flight Center Report, 1998.

[17] L. Rosenberg, "Metrics for object oriented environment," in *Proceedings of the EFAITP/AIE 3rd Annual Software Metrics Conference*, December 1997.

[18] K. Erni and C. Lewerentz, "Applying design-metrics to object-oriented frameworks," in *Proceedings of the 3rd International Software Metrics Symposium*, pp. 64–74, March 1996.

[19] R. Shatnawi, *The validation and threshold values of object-oriented metrics [Ph.D. dissertation]*, University of Alabama in Huntsville, Huntsville, Ala, USA, 2006.

[20] P. Louridas, D. Spinellis, and V. Vlachos, "Power laws in software," *ACM Transactions on Software Engineering and Methodology*, vol. 18, no. 1, pp. 1–26, 2008.

[21] R. Wheeldon and S. Counsell, "Power law distributions in class relationships," in *Proceedings of the 3rd IEEE International Conference in Source Code Analysis and Manipulation*, pp. 45–54, September 2003.

[22] M. E. J. Newman, "Power laws, Pareto distributions and Zipf's law," *Contemporary Physics*, vol. 46, no. 5, pp. 323–351, 2005.

[23] S. Valverde, R. F. Cancho, and R. V. Sol, "Scale-free networks from optimal design," *Europhysics Letter*, vol. 60, no. 4, pp. 512–518, 2002.

[24] S. Valverde and R. V. Sole, "Hierarchical small worlds in software architecture," *Dynamics of Continuous Discrete and Impulsive Systems B*, vol. 14, pp. 1–11, 2007.

[25] G. Concas, M. Marchesi, S. Pinna, and N. Serra, "Power-laws in a large object-oriented software system," *IEEE Transactions on Software Engineering*, vol. 33, no. 10, pp. 687–708, 2007.

[26] G. Baxter, M. Frean, J. Noble et al., "Understanding the shape of Java software," in *Proceedings of the ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages, and Applications*, pp. 397–412, New York, NY, USA, 2006.

[27] L. Hatton, "Power-law distributions of component size in general software systems," *IEEE Transactions on Software Engineering*, vol. 35, no. 4, pp. 566–572, 2009.

[28] C. Andersson and P. Runeson, "A replicated quantitative analysis of fault distributions in complex software systems," *IEEE Transactions on Software Engineering*, vol. 33, no. 5, pp. 273–286, 2007.

[29] A. Mubarak, S. Counsell, and R. Hierons, "Does an 80:20 rule apply to Java coupling?" in *Proceedings of the International Conference on Evaluation and Assessment in Software Engineering*, Keele, UK, 2009.

[30] M. Kaaniche and K. Kanoun, "Reliability of a commercial telecommunications system," in *Proceedings of the 7th International Symposium on Software Reliability Engineering (ISSRE '96)*, pp. 207–212, November 1996.

[31] G. Denaro and M. Pezze, "An empirical evaluation of fault-proneness models," in *Proceedings of the 24th International Conference on Software Engineering (ICSE '02)*, pp. 241–251, 2002.

[32] J. C. Munson and T. M. Khoshgoftaar, "The detection of fault-prone programs," *IEEE Transactions on Software Engineering*, vol. 18, no. 5, pp. 423–433, 1992.

[33] M. L. Goldstein, S. A. Morris, and G. G. Yen, "Problems with fitting to the power-law distribution," *European Physical Journal B*, vol. 41, no. 2, pp. 255–258, 2004.

[34] M. Mitzenmacher, "A brief history of generative models for power law and lognormal distributions," *Internet Mathematics*, vol. 1, no. 2, pp. 226–251, 2004.

[35] L. Etzkorn, C. Davis, and W. Li, "A practical look at the lack of cohesion in methods metric," *Journal of Object-Oriented Programming*, vol. 11, no. 5, pp. 27–34, 1998.

[36] Y. Zhou and H. Leung, "Empirical analysis of object-oriented design metrics for predicting high and low severity faults," *IEEE Transactions on Software Engineering*, vol. 32, no. 10, pp. 771–784, 2006.

[37] A. Clauset, C. R. Shalizi, and M. E. J. Newman, "Power-law distributions in empirical data," *SIAM Review*, vol. 51, no. 4, pp. 661–703, 2009.

[38] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge Press, Cambridge, UK, 2nd edition, 1992.

[39] H. M. Park, "Univariate analysis and normality test using SAS, stata, and SPSS," Working Paper, The University Information Technology Services (UITS) Center for Statistical and Mathematical Computing, Indiana University, 2008.

[40] T. Tamai and T. Nakatani, "Analysis of software evolution processes using statistical distribution models," in *Proceedings of the 5th International Workshop on Principles of Software Evolution (IWPSE '02)*, pp. 120–123, May 2002.

[41] "NIST/SEMATECH e-handbook of statistical methods," 2009, http://www.itl.nist.gov/div898/handbook/eda/section3/histogr6.htm.

[42] P. T. von Hippel, "Mean, median, and skew: correcting a textbook rule," *Journal of Statistics Education*, vol. 13, no. 2, 2005, http://www.amstat.org/publications/jse/v13n2/vonhippel.html.

[43] R. Shatnawi, "A quantitative investigation of the acceptable risk levels of object-oriented metrics in open-source systems," *IEEE Transactions on Software Engineering*, vol. 36, no. 2, pp. 216–225, 2010.

[44] R. Shatnawi, W. Li, J. Swain, and T. Newman, "Finding software metrics threshold values using ROC curves," *Journal of Software Maintenance and Evolution*, vol. 22, no. 1, pp. 1–16, 2010.

[45] M. Lanza and R. Marinescu, *Object-Oriented Metrics in Practice*, Springer, Berlin, Germany, 2006.

[46] L. A. Adamic, "Zipf, power laws, and Pareto—a ranking tutorial," Tech. Rep. 94304, Information Dynamics Lab, HP Labs, HP Labs, Palo Alto, Calif, USA, 2000.

[47] V. Pareto, "The new theories of economics," *Journal of Political Economy*, vol. 5, no. 4, pp. 485–502, 1897.

[48] L. Guo, Y. Ma, B. Cukic, and H. Singh, "Robust prediction of fault-proneness by random forests," in *Proceedings of the 15th International Symposium on Software Reliability Engineering (ISSRE '04)*, pp. 417–428, November 2004.

[49] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[50] M. M. Lehman, "Laws of software evolution revisited," in *Proceedings of the European Workshop on Software Process Technology*, pp. 108–124, October 1996.