On the Statistical Distribution of Object-Oriented System Properties

Israel Herraiz
Technical University of Madrid
Madrid, Spain
israel.herraiz@upm.es

Daniel Rodriguez
University of Alcala
Alcala de Henares, Spain
daniel.rodriguez@uah.es

Rachel Harrison
Oxford Brookes University
Oxford, United Kingdom
rachel.harrison@brookes.ac.uk

***Abstract*—The statistical distributions of different software properties have been thoroughly studied in the past, including software size, complexity and the number of defects. In the case of object-oriented systems, these distributions have been found to obey a power law, a common statistical distribution also found in many other fields. However, we have found that for some statistical properties, the behavior does not entirely follow a power law, but a mixture between a lognormal and a power law distribution. Our study is based on the Qualitas Corpus, a large compendium of diverse Java-based software projects. We have measured the Chidamber and Kemerer metrics suite for every file of every Java project in the corpus. Our results show that the range of high values for the different metrics follows a power law distribution, whereas the rest of the range follows a lognormal distribution. This is a pattern typical of so-called double Pareto distributions, also found in empirical studies for other software properties.**

***Keywords*-object-oriented properties; statistical distribution; Chidamber & Kememer; double Pareto; lognormal; power law; Qualitas Corpus**

Replication package: http://mat.caminos.upm.es/∼iht/wetsom2012/

I. Introduction

Statistical distribution research has addressed many different artificial phenomena, and researchers have found that power law distributions are a good description of empirical data for many of these phenomena. Software is not an exception [1], and this includes object-oriented systems [2], [3], [4]. However there is no consensus about whether software and object-oriented systems can be best described using power laws, or whether alternatives such as the lognormal distribution are better. Also, the reason these distributions appear in software is not yet clear. Some attribute this behavior to the network-like structure of object-oriented systems [4], but in fact power laws are found in many different kind of software systems [1], not only in object-oriented software.

The controversy about whether power laws describe the structure of software better than lognormal or other distributions is not exclusive to software systems, but also applies to other artificial systems where power laws seem to describe the statistical properties of the systems well [5], [6]. An area in which the issue has been deeply explored is file size. In that area, Mitzenmacher [7] found that a double Pareto distribution was a better fit than both the power law and lognormal distributions, and proposed a generative model that matches the software development process. In fact, the same distribution was found for the size of source code files, for many programming languages [8].

In the case of object oriented systems, although at a first glance some object oriented properties seem to follow power law distributions, the truth is that both lognormal and power law distributions have been reported for different properties, and the border between the two is narrow [2]. We believe this situation is exactly the same as has been previously found in other areas [5]; we try to offer new insights on the question by reporting our findings on a set of object-oriented software projects, after measuring the Chidamber and Kemerer (CK) metrics suite [9] for all of them. Our set of case studies was obtained from the Qualitas Corpus [10], and contains 69 open source projects written in Java. We have fitted lognormal, power law and double Pareto distributions for all the metrics, and have found that double Pareto is a better fit for most of the cases, which could explain the interplay between lognormal and power law distributions in object-oriented systems [2].

The rest of the paper is organized as follows. Section II describes the methodology (data used, metrics and statistical distributions used in the rest of the paper). Next, Section III enumerates the Java programs analyzed in this work. Section IV analyses the different distributions per metric followed by a summary (Section V and threats to validity in Section VI. Finally, Section VII concludes the paper and outlines future research work.

II. Methodology

A. Data Gathering

All the data used for this study was obtained from the Qualitas Corpus [10], which is publicly available. The Qualitas Corpus contains a set of 106 open source projects written in Java. The data includes source code, JAR files (compiled versions of the source code), documentation and meta-data about the project, with some basic metrics and a classification of the projects.

We used the Qualitas Corpus version 20101126, “r” release. Out of the 106 projects we could only measure and fit the distributions for 69 of the projects due to technical reasons.

B. Metrics

We measured the set of CK metrics [9] for every class in all the projects of the dataset.

To obtain these metrics we used the tool `ckjm` (Chidamber and Kemerer Java Metrics) by Spinellis[1]. This tool is able to gather the metrics from (compiled) Java class files and JAR compressed files. We applied the tool to each of the JAR files available in each of the projects in the Qualitas Corpus. Thus, we obtained the values of the metrics using the compiled Java bytecode instead of source code.

In detail, the metrics we gathered are the following:

- Weighted methods per class (WMC)
- Depth of inheritance tree (DIT)
- Number of children (NOC)
- Coupling between object classes (CBO)
- Request for a class (RFC)
- Lack of cohesion in methods (LCOM)

For the case of WMC, we assigned a weight of 1 to all the methods in a class, and so in this study WMC is also equal to the number of methods in a class.

C. Statistical Analysis

The identification of a power law tail in empirical data is a difficult task, because of the variability that is usually found in the large values. We use the fitting procedure suggested by Clauset *et al.* [11], which is based on maximum-likelihood estimation, and goodness-of-fit tests using the Kolmogorov-Smirnov distance. This process is fundamentally different to the procedure reported previously on power laws in object oriented systems [2]. This difference may be the cause of the discrepancy between what we report here and the results reported by Concas *et al.* [2].

The method we use is able to calculate not only the parameters of the power law, but also to identify the values that are not well described by the power law. The value, x_{min}, is used to split the empirical data into two sets. In the original method, the set of high values is fitted with a power law distribution, and the set of low values is ignored. However, we have modified the method to fit the set of low values to a lognormal distribution. If $x_{min} < 1,000$, the power law was fitted using the procedure for discrete data reported by Clauset. Otherwise, we assumed the variable was continuous.

The double Pareto distribution is formed by two power law tails and a lognormal body. The power law tails are found in the very low and very high values, and the lognormal body joins these two power law tails. In our case, for very low values, given the kind of measurements we are dealing with, it is difficult to obtain a power law for the low value end, as the data are discrete, and the lognormal behavior starts with relatively low values. This problem (identifying the low values power law tail in double Pareto distributions) is similar to the case of file sizes, as reported by Mitzenmacher [7]. Therefore, taking this into account, and also considering that the generative model for double Pareto fits the software development process well, we estimate that all the distributions we studied with a power law tail in the high value end, and a lognormal distribution in the rest of values, are double Pareto distributions.

For the lognormal distribution part we fitted the distribution using the Kolmogorov-Smirnov distance as a measurement of the goodness of fit, because for the low value side there are no problems with the variability of the data. The fitting procedure is standard using maximum likelihood estimation.

For the fitting procedure and to obtain the plots shown in this paper we used the MATLAB programs provided by Clauset *et al.*[2]. We measured and fitted each project in a separate process in the CeSViMa's Magerit supercomputer. For the MATLAB fitting programs, we used GNU Octave 3.4.2 [12], and for the plot programs we used MATLAB R2011b on a desktop computer, using the MAT files obtained with GNU Octave. The fitting procedure recommended by Clauset *et al.* and implemented in the MATLAB programs is very expensive in terms of computing resources, and some projects took more than 24 hours to be successfully fitted. To obtain all the results reported in this paper, we consumed more than 1000 computation hours in Magerit.

D. Replication of This Study

The results of this study can be replicated thanks to the replication package[3], which includes information about the data and necessary scripts to run the experimental work discussed.

III. Cases Under Study

We initially selected all the 106 projects contained in the Qualitas Corpus. We applied the `ckjm` tool to all the JAR files found in each project. The tool provided no output for 37 of the projects and so these were discarded from our study. As this is a preliminary report, we did not investigate why the tool was not able to provide an output for those projects but we will do so in future work.

Table I shows the finally selected projects, including name and studied version, the domain of application, the size in SLOC (that is, removing comments and blank lines) and the number of classes (measured as the number of class files in the deployed JAR files). These data were obtained directly from the metadata included in the Qualitas Corpus.

[1] Available at http://www.spinellis.gr/sw/ckjm/

[2] Available at http://tuvalu.santafe.edu/∼aaronc/powerlaws/

[3] http://mat.caminos.upm.es/∼iht/wetsom2012/

Table I
Projects under study

System	Domain	SLOC	#Classes	System	Domain	SLOC	#Classes
ant-1.8.1	build	107770	1268	ivatagroupware-0.11.3	middleware	23786	381
antlr-3.2	build	25243	531	jFin_DateMath-R1.0.1	SDK	4807	62
aoi-2.8.1	graph	111725	863	jag-5.0.1	tool	14762	338
argouml-0.30.2	visualization	194859	2905	james-2.2.0	tool	27003	340
aspectj-1.6.9	progr. lang.	412394	2665	jasml-0.10	tool	5732	49
axion-1.0-M2	database	23744	261	jasperreports-3.7.3	visualization	170064	1844
azureus-4.5.0.4	database	453433	7249	javacc-3.2	build	13807	132
c_jdbc-2.0.2	database	81306	586	jboss-5.1.0	J2EE server	281643	15247
castor-1.3.1	middleware	115543	1663	jchempaint-2.0.12	SDK	6321	703
cayenne-3.0.1	database	127529	2184	jedit-4.3.2	tool	107469	1128
checkstyle-5.1	IDE	23316	352	jena-2.5.5	middleware	89987	1564
cobertura-1.9.4.1	testing	51860	122	jext-5.0	visualization	26565	504
colt-1.2.0	SDK	38625	593	jfreechart-1.0.13	tool	98078	857
columba-1.0	tool	71680	1335	jgraph-5.9.2.1	tool	12341	90
displaytag-1.2	visualization	11832	131	jgraphpad-5.10.0.2	tool	23750	431
drawswf-1.2.9	graph	27008	319	jgrapht-0.8.1	tool	11931	255
drjava-stable-20100913-r5387	IDE	62380	3877	jgroups-2.6.2	tool	85243	1033
eclipse_SDK-3.6	IDE	2282511	32126	jhotdraw-7.5.1	graph	75958	1070
emma-2.0.5312	testing	25806	330	jmeter-2.4	testing	81010	2077
findbugs-1.3.9	testing	109096	1744	jmoney-0.4.4	tool	8197	193
fitjava-1.1	testing	2240	61	joggplayer-1.1.4s	graph	14936	194
fitlibraryforfitnesse-20100806	testing	27539	1290	jparse-0.96	build	12559	69
freecol-0.9.4	games	81671	1077	jpf-1.5.1	SDK	13246	189
freecs-1.3.20100406	tool	23012	147	jrat-0.6	testing	14146	250
galleon-2.3.0	graph	52653	809	jre-1.6.0	progr. lang.	914867	17348
ganttproject-2.0.9	tool	47051	1058	jrefactory-2.9.19	tool	113427	1553
gt2-2.7-M3	SDK	446863	5613	jruby-1.5.2	progr. lang.	160360	5068
heritrix-1.8.0	tool	47272	531	jsXe-04_beta	tool	8829	107
hibernate-3.6.0-beta4	object mapper	163858	2674	jspwiki-2.8.4	middleware	43326	455
hsqldb-2.0.0	database	123268	535	jung-2.0.1	visualization	37989	858
htmlunit-2.8	testing	40004	932	junit-4.8.2	testing	6164	209
informa-0.7.0-alpha2	middleware	9722	170	log4j-1.2.16	testing	20637	308
ireport-3.7.5	visualization	221490	3394	marauroa-3.8.1	games	13823	227
itext-5.0.3	visualization	76369	544	picocontainer-2.10.2	middleware	9259	255
				sablecc-3.2	build	28394	286

IV. Results

A. Weighted Methods per Class (WMC)

As an example, Figure 1 shows the complementary cumulative distribution function (CCDF) for the WMC metric of the ArgoUML project (Power law parameters $x_{min} = 28$, $\alpha = 2.72$, Lognormal parameters $\mu = 1.61$, $\sigma = 0.99$). The plots shows the empirical estimation using the data, and two fits, the power law and lognormal models. The power law fit has been calculated only for values higher than a given threshold, which we call x_{min}. The lognormal model has only been fitted for values lower than this threshold.

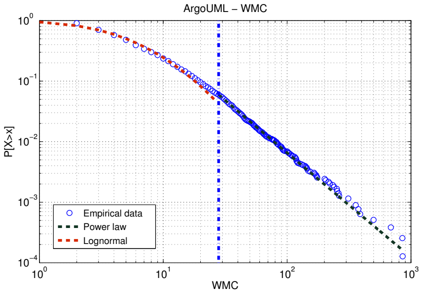

Figure 1. CCDF of the WMC metric for ArgoUML. The value of x_{min} is marked with a vertical line. Both axes are on a logarithmic scale

In a CCDF plot, the maximum vertical distance between two functions is called the Kolmogorov-Smirnov distance, D, and is used to evaluate the best fit between a set of possible models. In the case of the WMC metric, the lognormal model alone cannot explain the whole range of values, because it deviates from the empirical data for very large values, causing a larger value of D. Figure 2 shows this fitting. Note the evident power law tail which causes the large D value for the lognormal model alone. It is also worth noting that the deviation starts at x_{min}, which marks the border between the lognormal and power law sides in the double Pareto model.

On the other hand, the power law model cannot explain all the values either, because it deviates from the empirical data for small values, causing again a larger value D. However, a hybrid model can explain the whole range of values with minimal deviation from the empirical data, that is, with a lower value of D.

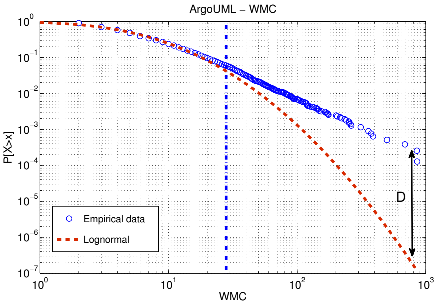

Figure 2. CCDF of the WMC metric for ArgoUML, with only the lognormal model fitted. Both axes are on a logarithmic scale

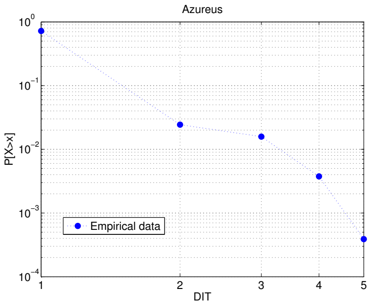

Figure 3. CCDF of the DIT metric for Azureus. Both axes are on a logarithmic scale

This is the typical behavior of a double Pareto distribution, although with a small modification: we are missing the power law tail for the very low values. The WMC metric is of course a discrete variable. For very low values we could not fit a power law neither a lognormal. We attribute this difficulty to the influence of noise for very low values. This influence is negligible for very large values though. When we fit the lognormal body, we should discard very low values, because they are probably not distributed lognormally. However, the difference in the empirical data and the lognormal model including those values is very small. This is probably the cause of the small difference we see around the value of x_{min} between the two fits.

We do not show the rest of plots due to the lack of space in this paper, but the same behavior was verified with all the projects using the WMC metric.

Concas *et al.* [2] have previously reported that the WMC metric follows a lognormal distribution, although they admit that a power law could also be fitted if they discard very low values. As we do here, they chose to assign a weight of 1 to all the methods. So our data are comparable with theirs. We believe that the data reported by Concas *et al.* for the WMC metric is a double Pareto model, which would explain the good power law fit of the high values tail, and also the good fit of the lognormal model. In our case, fitting a lognormal model alone is not as accurate as the double Pareto model.

We note that our fit procedure is not fitting a *combined* double Pareto model, but we fit the power law for the range of high values, and the lognormal for the range of low values. We then plot everything together. We calculate the threshold value using the fitting procedure recommended by Clauset *et al.* [11].

B. Depth of Inheritance Tree (DIT)

Figure 3 shows the CCDF estimation for the DIT metric of the Azureus project. We have added a line to join the dots for more clarity. As the plot shows, there are very few possible values for the DIT metric, which makes it very hard to fit a model to the empirical data. The plots are similar for the rest of case studies in our sample.

In fact, this metric is not reported in other similar studies about the power law nature of object oriented properties [2], [3], [4].

C. Number of Children (NOC)

Figure 4 shows the plots of the CCDF of the NOC metric for the projects Eclipse SDK ($x_{min} = 3$, $\alpha = 2.25$), Findbugs ($x_{min} = 8$, $\alpha = 2.49$), Freecol ($x_{min} = 28$, $\alpha = 3.07$) and ArgoUML ($x_{min} = 2$, $\alpha = 2.16$). The plots show the empirical data and the power law fitted from the x_{min} threshold onward.

It seems clear that the behavior of NOC is different to WMC. The case of Eclipse SDK shows that the NOC metrics seem to follow a power law, without any kind of lognormal part. The case of Findbugs is similar, although in this case our fitting procedure could not fit the low values side, that is, the x_{min} value in the case of Eclipse SDK is very low, indicating a pure power law, but it is not as low in the case of Findbugs. We could not fit a lognormal model or a power law for the low values part in the case of Findbugs. In the case of the power law, repeating the fitting using the same procedure but with all the values lower than x_{min}, our procedure obtained a new threshold value for the low values which was very close to the x_{min}, meaning that the power law was very "short", and in essence could not be fitted to the data.

We found another interesting pattern in the case of Freecol. The behavior is similar to Findbugs, but the difference between the low values and the high values is even clearer. This time, at a first glance the CCDF seems to be divided in two straight lines, which are therefore power laws. We could fit a power law for high values, but when we tried to fit a power law to the low values the value of x_{min} was very close to the threshold of the high values, meaning that the power law could not be successfully fitted. We also attempted to fit a lognormal model, again without success.

We attribute the difficulties to fitting a power law in the case of Freecol and Findbugs to our fitting procedure, which is suitable in the case of high variability of large values, but

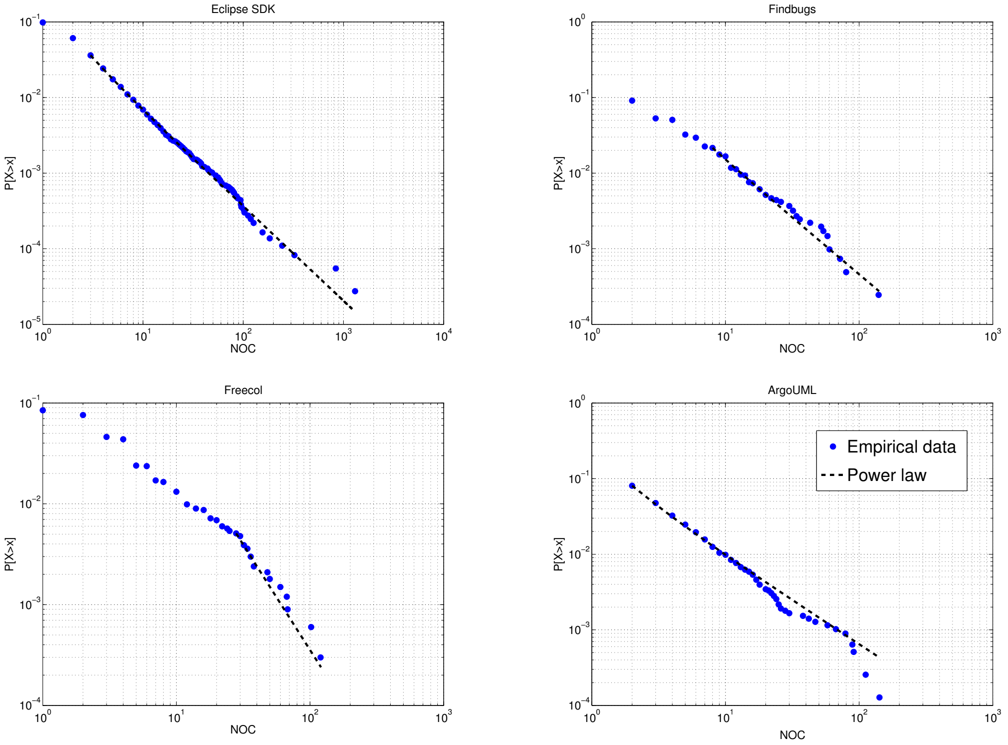

Figure 4. CCDF plots for the NOC metric of Eclipse SDK, Findbugs, Freecol and ArgoUML

not for the case of very low values, that don't have a high variability.

The three projects, Eclipse SDK, Freecol and Findbugs, presented the same profile in the case of the WMC metric, that is, a double Pareto, with a power law tail for the large values and lognormal for the low values. To complete the picture, Figure 4 also shows the CCDF plot for the NOC metric of ArgoUML, which we also found for WMC. In this case, the NOC for ArgoUML seems to follow a pure power law, as in the case of Eclipse SDK.

D. *Coupling Between Classes (CBO)*

Figure 5 (next page) shows the CCDF plots for the CBO metric of Hibernate ($x_{min} = 18$, $\alpha = 3.5$, $\mu = 1.30$, $\sigma = 0.98$) and HSQLdb ($x_{min} = 23$, $\alpha = 3.5$, $\mu = 1.67$, $\sigma = 1.11$). Again, as in the case of WMC, we find that the CCDF can be divided into two parts, one that is better described by a lognormal distribution (low values) and a power law tail (high values). In the case of HSQLdb, the shape is not as clear as for Hibernate, because the power law tail has a sudden cutoff for very large values.

We also attempted to fit only a lognormal distribution for the whole range of values, again with higher Kolmogorov-Smirnov distances than in the case of the double Pareto distribution, so it is a worse fit than the case of the double Pareto distribution.

E. *Request for a Class (RFC)*

Figure 6 shows the CCDF plot of the RFC metric for the JEdit system ($x_{min} = 73$, $\alpha = 3.04$, $\mu = 2.54$, $\sigma = 1.10$). The behavior is similar to that of WMC and CBO.

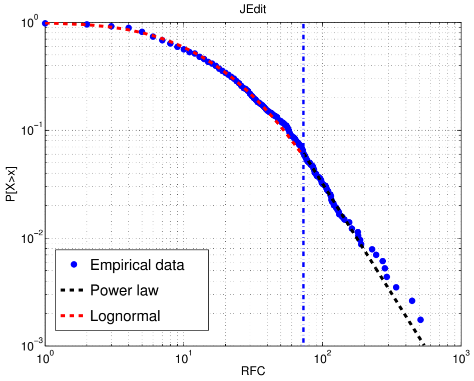

Figure 6. CCDF plots for the RFC metric for JEdit

F. *Lack of Cohesion in Methods (LCOM)*

Figure 7 shows the CCDF plot of the LCOM metric for the JUnit system ($x_{min} = 6$, $\alpha = 1.62$). On this occasion, the behavior is similar to the NOC metric. The value of the x_{min} threshold is very low, so almost the whole range of values is covered by a power law. We attempted to fit

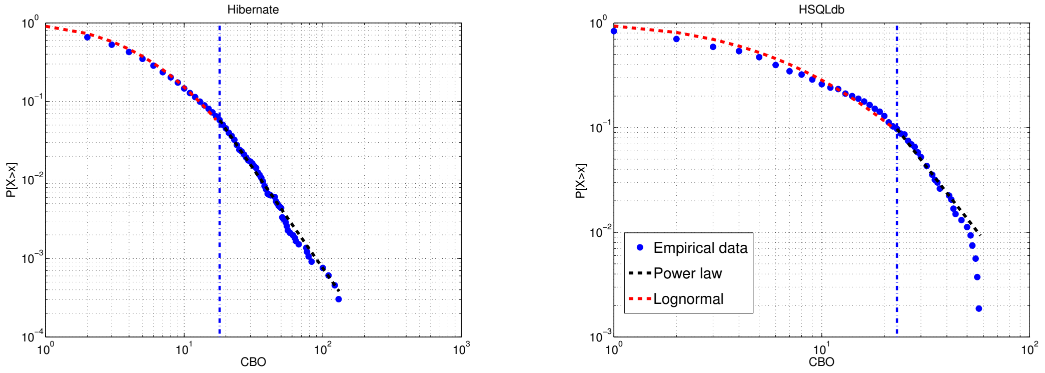

Figure 5. CCDF plots for the CBO metric of Hibernate and HSQLdb

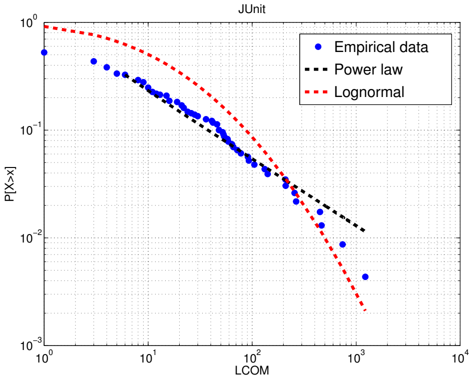

Figure 7. CCDF plots for the LCOM metric for JUnit

a lognormal distribution to the whole range as well, and the result is shown also in Figure 7. Please note that this lognormal fit is not similar to the previous cases. That is, we are not combining a lognormal and a power law in this case. We attempted to fit the lognormal because there is an observable deviation from the power law fit for very large values, to find out whether the lognormal model could explain the data better. But although the tail for large values seems to be well described by the lognormal fit, the Kolmogorov-Smirnov distance is higher in the case of the lognormal fit than in the power law, and therefore the power law is the most likely model. We repeated the same procedure for the other systems, and the power law fitting was even clearer in most cases.

V. Summary of Results

We can classify the suite of CK metrics into three categories, according to their statistical properties:

- Metrics better described by a double Pareto (lognormal plus power law) distribution
- Metrics better described by a sole power law distribution
- Metrics that cannot be fitted to either a lognormal or a power law

For the first case, double Pareto metrics, we have found that **weighted methods per class (WMC, with weights equal to 1), coupling between object classes, and requests for a class (RFC) are all better described using a double Pareto**, with a power law tail for large values and a lognormal distribution for values lower than a threshold x_{min}.

For the second case, power law metrics, we have found that **number of children (NOC) and lack of cohesion in methods (LCOM) are better described using a power law** for the whole range of values.

For the third case, the **depth of inheritance tree (DIT) distribution could not be described using either a lognormal or a power law**.

VI. Threats to Validity

External validity. Our study is based only on projects extracted from the Qualitas Corpus. Moreover, for technical reasons, we could not measure some of the systems. We believe that there is no reason to think that using the Qualitas Corpus inserts a bias in the results, and neither does the fact that we are only considering Java projects.

Internal validity. For the statistical analysis, we have to make sure that the distributions we found are statistically significant, that is, we can statistically reject the null hypothesis that the data are random. This is a preliminary report and we have not been able to calculate p values for the estimated parameters, due to the excessive computation time needed. However, our procedure can obtain such values, and we are now in the process of calculating and including them in a next version of this report (some of them are available in the replication package). In any case, we have used the Kolmogorov Smirnov distance as a measurement of goodness of fit, and in all the cases the values are small and the empirical and estimated CCDF are very similar. However, we need to extend the study and test the hypothesis that the data is not random.

Construct validity. We are only studying one release for each of the systems. The shape of the distribution could change over time with different releases, due for instance to different programming and maintenance practices between different releases. If that were the case, the shapes found in this paper could not be attributed to the object oriented metrics, but to other factors. We need to explore more releases of the same systems to discard other cofactors. We are also trusting the tool used to measure the CK metrics suite, and we have not manually validated the results. although the `ckjm` tool has been also used in other work (e.g. [13]) and is trusted, differences in the measurement values have been reported [14]. We plan to investigate the results with other tools, specially if there are differences between tools using the source or compiled code as is the case of `ckjm`.

VII. Conclusions

Power laws have been termed the *signature of human activity*, as they have been found in many different areas and in many processes of human origin. However, whether many of these properties are a power law or a lognormal distribution remains an open question.

In the case of object oriented systems, for the CK metrics suite, previous research found that some of the metrics were power law distributed, and some other where lognormally distributed.

However, we have found that for most of the CK metrics, there is a model that can describe the data better: a double Pareto distribution. The same distributions have been found in file-systems and source code file sizes..

We have extended the double Pareto finding to the case of object oriented properties, an area where the debate between the power law and lognormal distributions remains open. Our study is based on publicly available data and is easily repeatable and verifiable thanks to the provided replication package. Our results show that some of the CK metrics are better described using a double Pareto distribution, although some of the metrics are better described by a power law distribution. We have not yet explored the reasons for this difference. We plan to explore the influence of other cofactors (such as software size and complexity, domain of application, and information about the development and maintenance process), using data available in the Qualitas Corpus, to try to explain this difference and to find out if the double Pareto parameters are related to other system properties.

Acknowledgments

The authors thankfully acknowledge the computer resources, technical expertise and assistance provided by the Centro de Supercomputación y Visualización de Madrid (CeSViMa), of the Universidad Politécnica de Madrid.

References

[1] P. Louridas, D. Spinellis, and V. Vlachos, “Power laws in software,” *ACM Transactions on Software Engineering and Methodology*, vol. 18, pp. 2:1–2:26, October 2008. [Online]. Available: http://doi.acm.org/10.1145/1391984.1391986

[2] 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.

[3] Y. Yi, H. Song, R. Zheng-ping, and L. Xiao-ming, “Scale-free property in large scale object-oriented software and its significance on software engineering,” in *Information and Computing Science, 2009. ICIC ’09. Second International Conference on*, vol. 3, may 2009, pp. 401 –404.

[4] R. Wheeldon and S. Counsell, “Power law distributions in class relationships,” in *Source Code Analysis and Manipulation, 2003. Proceedings. Third IEEE International Workshop on*, sept. 2003, pp. 45 – 54.

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

[6] ——, “Editorial: The future of power law research,” *Internet Mathematics*, vol. 2, no. 4, pp. 525–534, 2005.

[7] ——, “Dynamic models for file sizes and double Pareto distributions,” *Internet Mathematics*, vol. 1, no. 3, pp. 305–333, 2004.

[8] I. Herraiz, D. German, and A. E. Hassan, “On the distribution of source code file sizes,” in *International Conference on Software and Data Technologies*, Seville, Spain, 2011.

[9] S. R. Chidamber and C. F. Kemerer, “A metrics suite for object oriented design,” *IEEE Transactions on Software Engineering*, vol. 20, no. 6, pp. 474–493, June 1994.

[10] E. Tempero, C. Anslow, J. Dietrich, T. Han, J. Li, M. Lumpe, H. Melton, and J. Noble, “Qualitas corpus: A curated collection of java code for empirical studies,” in *2010 Asia Pacific Software Engineering Conference (APSEC2010)*, Dec. 2010.

[11] A. Clauset, C. R. Shalizi, and M. E. J. Newman, “Power-law distributions in empirical data,” 2007. [Online]. Available: http://www.citebase.org/abstract?id=oai:arXiv.org:0706.1062

[12] J. W. Eaton, *GNU Octave Manual*. Network Theory Limited, 2002.

[13] J. Singer, G. Brown, M. Luján, and I. Watson, “Towards intelligent analysis techniques for object pretenuring,” in *Proceedings of the 5th international symposium on Principles and practice of programming in Java*. ACM, 2007, pp. 203–208.

[14] R. Lincke, J. Lundberg, and W. Löwe, “Comparing software metrics tools,” in *Proceedings of the 2008 international symposium on Software testing and analysis*. ACM, 2008, pp. 131–142.