On Comparing Software Quality Metrics of Traditional vs Blockchain-Oriented Software: An Empirical Study

Marco Ortu
University of Cagliari
Cagliari, Italy
marco.ortu@diee.unica.it

Matteo Orrú
Universitá degli Studi di Milano-Bicocca
Milano, Italy
matteo.orru@unimib.it

Giuseppe Destefanis
Brunel University
London, United Kingdom
giuseppe.destefanis@brunel.ac.uk

***Abstract*—Driven by the surge of interest generated around blockchain technologies over the last years, a new category of systems, called Blockchain-Oriented Software (BOS), which are strictly tied to Blockchain distributed environment, has become increasingly popular. Yet, there is not a thorough understanding of their structure and behaviour and if and to which extent they differ from traditional software systems. The present work provide a first statistical characterisation of BOS. We analysed and compared 5 C++ open source Blockchain-Oriented and 5 *Traditional* Java software systems, aiming at detecting potential differences between the two categories of projects, and specifically in the statistical distribution of 10 software metrics. Although, in general, the statistical distributions for Traditional software and Blockchain software show similarities, the distribution of Average Cyclomatic and Ration Comment To Code metrics reveal significant differences in their queue, whereas the Number of Statements metric shows meaningful differences on the double Pareto distribution.**

***Index Terms*—Mining software repositories, metrics, blockchain oriented software, software engineering**

I. Introduction

During the last years, we have been witnessing an increasing interest in blockchain technologies both from industry and academy.

Designed in the first place as the backbone architecture for safely and securely managing and tracking Bitcoin transactions [18], the blockchain architecture crossed the borders of the realm of cryptocurrencies to become a core element of an increasing number of technologies (i.e. smart-contract, etc.) and it is now exploited in a high number of application domains.

Driven by the success of the blockchain technology, developers started designing and programming software in a different, blockchain-oriented way, to the point that a new category of software system emerged. Blockchain-oriented Software (BOS) [22] is characterized by the fact that it interacts with the distributed environment represented by a blockchain, which imposes some constraints and new challenges to developers.

As it happens for regular software systems, a relevant issue is that of assuring a sustainable development process which leads to a final product as much as possible reliable and bug-free or, in other words, a high-quality product. To achieve these goals, traditional software engineering practices look at software metrics as a mean to keep under control the software development – "You can't control what you can't measure", according to the famous quote from De Marco [7].

The study of software metrics for assessing software quality is a well known and sound discipline, starting from the early days of the Object-oriented paradigm adoption, with the suite proposed by Chidamber and Kemerer [5], and today there are a number of popular metrics for software evaluation (either object-oriented or not). Even if software metrics suites are reasonably used in practice, today we still lack a theoretically sound reason for the adoption of one metrics suite or another.

This is probably because the term *quality* can be subject to interpretation and can be declined in different ways. Additionally, it is often difficult to correlate software quality, as whole concepts, to the set of available metrics. To make things even harder to deal with, specifically regarding BOS systems, a new paradigm is emerging which lead to re-thinking the way software system are designed and delivered.

Practitioners and developers need to be supported by scientists in the decision of which metrics suite better fits their needs. This decision can be even more critical in the case of BOS systems. This work joins the efforts of clarifying this point, in the specific context of blockchain oriented software systems. The first step is to provide a statistical characterisation of blockchain oriented software system, to shed some light on their *shape* [2] structure and behaviour.

The structure of a software system designed to deal with the blockchain is still unknown. Moreover, we are not aware of any study which provides a statistical characterisation of the properties of such systems. Contrarily to what happened for software system written in Java [8], [10] no studies are dealing with the study of the distribution of traditional metrics in blockchain-oriented software systems.

The present work introduces the results of an analysis of the statistical distribution of 9 traditional metrics performed on a total of ten software systems (five classified as "Traditional" and five classified as "Blockchain-oriented"), with the aim of highlighting potential differences between the former kind of systems and the latter. In this paper, we followed the

methodology used by Destefanis et al. [10], and replicated the experiment considering Blockchain and traditional software systems (instead of Java vs Python).

The rest of this paper is structured as follows: In Section II, we provided some background information to establish a common language concerning some related work. Section III illustrates the methodology adopted in this work. In Section IV, we presented and discussed the results whereas in Section V we outlined the threats to validity. Finally, in Section VI we presented some final remarks.

II. Background

Several studies on the distribution of software metrics share the same goal of providing a way to improve the software development process. Some of them show that there is a correlation with the known Chidamber and Kemerer (CK) suite [6] and software fault proneness and maintenance issues [4], [26], [29]. Destefanis et al. [10] used a suite of object-oriented metrics to analyse differences between Java and Python programming languages show that it is possible to distinguish important differences in metrics distributions. In this study, we followed the same procedure used by Destefanis et al. [10], but replicating the study considering Blockchain vs traditional software systems. Micro-patterns were also considered and used fro providind insights to developers and managers in better understanding and maintaining software quality [11], [12], [19], [20].

According to Alshayeb et al. [1] Object Oriented metrics show predictive power in the agile context, with some caveats. The authors demonstrated that OO metrics could be used to effectively predict both design efforts and changes in the source code in the short term, although they proved to be rather ineffective in the long term.

Potanin et al. [23] showed that the networks associated with object-oriented software written in different languages are scale-free.

Louridas et al. [15] reported a pervasive presence of long fat tail statistical distributions in software systems. The authors show that such statistics show up in different contexts and at different levels of abstraction.

Shatnawy et al. [24] have shown that the power law behaviour is prevalent. The authors studied 5 Open Source systems with the aim of investigating the distribution and improving the fault prediction models. The authors found that, provided a given reasonable threshold, the properties characterised by a power law distribution might help improving fault prediction models. The threshold itself can be derived by the power law distribution.

Software metrics might follow different statistical distributions, with different ranges of variations. A case in point it the Depth of inheritance tree (DIT) which typically is lower than 10. This makes them less interesting for us due to the fact the associated statistics cannot be used to distinguish a BOS system from a regular one. On the other hand, we focus our attention on metrics which have a full range of variation, because it makes it easier to spot the difference between the two software systems. An approximately power-law distribution might be fitted typically by many different statistical distributions. We select the candidate distribution which can guarantee the best fitting but also were able to reveal meaningful differences among the chosen metrics. For this reason, we consider the log-normal and the double Pareto distribution.

The double Pareto extends the standard power-law, where two different power-law regimes co-exist and are characterised by the following two parameters α and β in eq. 1. In the literature we can find a variety of double Pareto distribution [17], [25]. For the purpose of the present work, we are going to use the form described in [25] for the Complementary Cumulative Distribution Function (CCDF). The reason behind this decision is related to the fact that it provides a smoother transition across the two different power-law regimes:

$$P(x) = 1 - \left[\frac{1 + (m/t)^{-\alpha}}{1 + (x/t)^{-\alpha}}\right]^{\beta/\alpha} \quad (1)$$

The double Pareto distribution is flexible enough to fit a power-law tail distribution at the head of the distribution, where it is very similar to a log-normal. This fact makes it more effective than a single power-law for fitting purposes.

Researchers have focused on software engineering aspects of Blockchain software [3], [9], [13], [14], [16], [21], [22], [27], [28], and they found out that "Blockchain" software has several differences from "Traditional" software.

III. Methods and Materials

This study aims to compare metrics of "traditional" and "blockchain oriented" software; thus we have selected five "traditional" Java projects and five C++/go *blockchain* software, a total of ten projects for our analysis.

We have considered systems from GitHub.com [1]: Table I shows the software systems selected. We can can distinguish the "Traditional" (in **bold**) and Blockchain software.

TABLE I
Blockchain and Traditional Projects

Blockchain	**Traditional**
Bitcoin-core	**Apache Cassandra**
Ehtereum-go	**Apache Hadoop**
Monero	**Pentaho Platform**
Dodgecoi	**Zookeeper**
Ripple	**Tomcat**

In order to extract software metrics, we used Understand 3.1 (build 766)[2], and selected nine metrics for our analysis:

- Number of base classes
- Number Of Declared Instance Methods
- Number of Declared Instance Variables
- Number Of Local Methods
- Number of Lines of code
- Number of Lines of comments

[1] https://github.com/

[2] Understand. Scitools.com: https://scitools.com

- Number of statements
- Average Cyclomatic
- Ration Comment To Code

We extracted the empirical cumulative distribution function (ECDF) of each of the ten systems and fit the Log-normal and the double Pareto distributions to check possible statistical differences among metrics. We performed statistical analysis with R[3] and Matlab[4].

Out of the nine metrics selected we focused on:

- Average Cyclomatic (AC)
- Ration Comment To Code (RCC)
- Number Of Statements (NOS)

These metrics provide the most significant results in our analysis. As a first step, we used a best fitting procedure on the metric distribution using the douple Pareto and the log-normal distribution functions.

We used the goodness of fit parameter provided by Kolmogorov-Smirnov statistic: KV. We have systematically computed KV for each system. The closer to zero the goodness of fit coefficient is, the better. The KV coefficient is interpreted as follows:

- close to 0, the fit is considered to be very good.
- larger than 0.6, the fit is considered fairly statistically good.
- values greater than 0.6 provide a worse fit.

The best fitting parameters values of the ten analysed projects statistically represent the metrics values for every system considered in the study. The parameters are μ and σ for the log-normal distribution, and α and β for double Pareto distribution. We confronted the two sets of parameters to find out statistically differences between the Traditional projects and Blockchain projects.

IV. Results

The majority of the empirical distributions obtained from the metrics analysis are at first look very similar.

However, three metrics showed that differences between Blockchain and Traditional projects could be detected:

- the Average Cyclomatic (AC), Average cyclomatic complexity for all nested functions or methods (AC) ;
- the Number of declarative statements (NOS);
- the Ratio of Comment lines to Code lines (RCC);

We first analysed the results obtained with the AC metric. The AC Empirical Cumulative Distribution Function (ECDF) shows that the Log-normal distribution fits for Blockchain and Traditional software only for low values, as the goodness-of-fit coefficient KS is 0.236 and 0.190 respectively, indicating a better fit for Traditional software. Different averages for the location parameter are provided by the fitting procedures highlighting a difference between the two sets of projects, and with Blockchain projects presenting a significantly higher value. Table II presents the comparison among average values obtained for the Log-normal distribution parameters.

[3]The R Project for Statistical Computing - www.r-project.com
[4]www.mathworks.com

TABLE II
AC CCDF Averaged Log-normal fit parameters with standard deviation

	μ	σ	KV
Blockchain	0.656	0.952	0.366
Traditional	0.611	1.373	0.401

TABLE III
AC CCDF Averaged double Pareto fit parameters with standard deviation

	α	β	KV
Blockchain	0.532	1.129	0.28
Traditional	0.475	0.621	0.366

Following the methodology used by Destefanis et al. [10], we performed the rank sum test and the Kruskal-Wallis statistical test on the obtained values of μ and of σ (five for each category) to verify if the two sets exhibit statistical differences among their metrics.

For the log-normal distribution, both tests provide a positive answer. Table III shows that the parameters μ and σ, obtained from the best fitting with a log-normal distribution for the metric AC permit to differentiate between Blockchain and Traditional systems. The position parameters present significant higher values for Blockchain projects, with a maximum value of 500 against a value of 20 for Traditional projects (see fig.1 and 2).

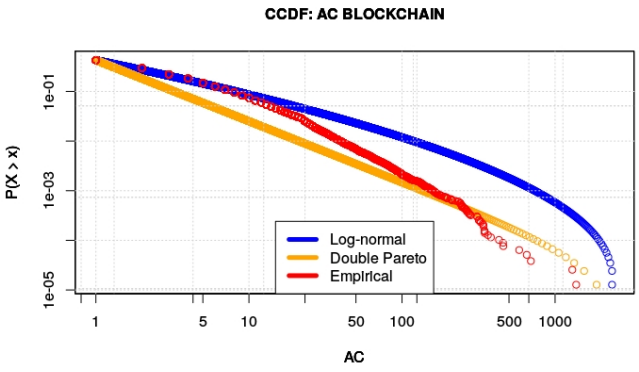

Fig. 1. AC Blockchain-oriented Software.

Table IV shows the average values for α and β best fitting parameters calculated for the metric NOS. The coefficient of determination KV is close to zero, indicating a good fit with the double Pareto distribution, and again the fitting is better for Traditional software. Table V shows that the averages of the parameter β obtained from Blockchain systems differ from the Traditional systems.

Parameter α provides the power-law exponent in the first power-law regime, while the parameter β provides the power-law exponent for the second regime. The results show that significant differences are evident along the queue of the distributions, specifically for higher values of the NOS metric. The use of the Number of Declarative Statements in Blockchain

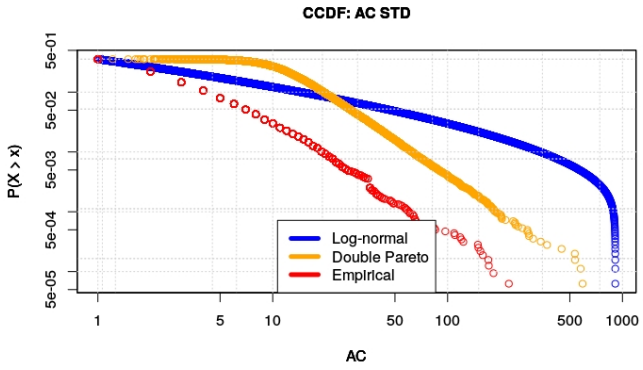

Fig. 2. AC Traditional Software.

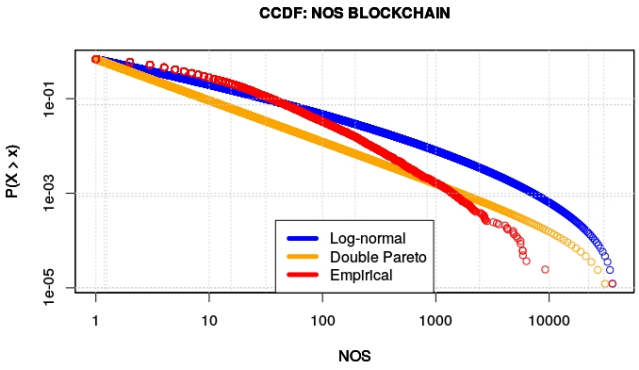

Fig. 3. NOS Blockchain-oriented Software.

and Traditional projects is different. A difference between metric distributions has been found, and again we can see from figures 5 and 6 Blockchain software has a relatively larger range of values with a maximum of 10000 against 50 for Traditional.

Table IV reports the same results for the best fitting parameters of the log-normal distribution concerning the metric NOS.

TABLE IV
NOS CCDF AVERAGED LOG-NORMAL FIT PARAMETERS WITH STANDARD DEVIATION

	μ	σ	KV
Blockchain	1.539	1.466	0.347
Traditional	1.75	1.63	0.375

TABLE V
NOS CCDF AVERAGED DOUBLE PARETO FIT PARAMETERS WITH STANDARD DEVIATION

	α	β	KV
Blockchain	0.552	2.734	0.158
Traditional	1.01	5.83	0.192

The log-normal distribution provides a better fitting for the NOS metric CCDF than the previous case, being the coefficient of determination below 0.19 for Traditional software. Nevertheless, the analysis suggests that the related dispersion parameter can be used for distinguishing Blockchain projects from Traditional projects.

In figures 5 and 6 we plotted the RCC CCDF and related log-normal fit for Blockchain and Traditional projects applying the best fitting procedure for the RCC CCDF and using the double Pareto statistical distribution. The obtained results are similar to those obtained with the log-normal, concerning the coefficient of determination. However, the results are not useful for distinguishing between Blockchain and Traditional projects. The results for the averages of the best fitting parameters and the coefficient of determination for the double

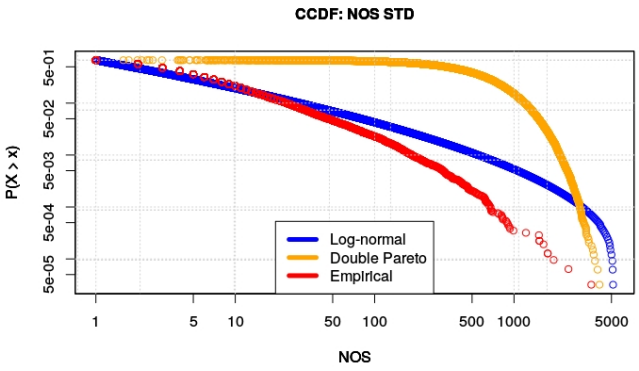

Fig. 4. NOS Traditional Software

Pareto case for the RCC metric, as well as the relative standard deviations, are presented in table V.

Finally, RCC CCDF show more significant differences between Blockchain and Traditional projects than RCC CCDF (see figure 6).

The average value for the coefficient of determination is very high for both sets of projects. The log-normal location parameters are dissimilar: the average value for Blockchain systems is around 0.364, while for Traditional is approximately 1.14. Mean and standard deviation for Blockchain and Traditional projects are presented in table VI.

TABLE VI
RCC CCDF AVERAGED LOG-NORMAL FIT PARAMETERS WITH STANDARD DEVIATION

	μ	σ	KV
Blockchain	0.656	0.952	0.366
Traditional	0.611	1.373	0.401

The obtained results with the double Pareto distribution, show that the parameter α permits to differentiate between the two sets of projects, while the parameter β is not helpful, as shown in table VII.

TABLE VII
RCC CCDF AVERAGED DOUBLE PARETO FIT PARAMETERS WITH STANDARD DEVIATION

	α	β	KV
Blockchain	0.532	1.129	0.28
Traditional	0.475	0.621	0.366

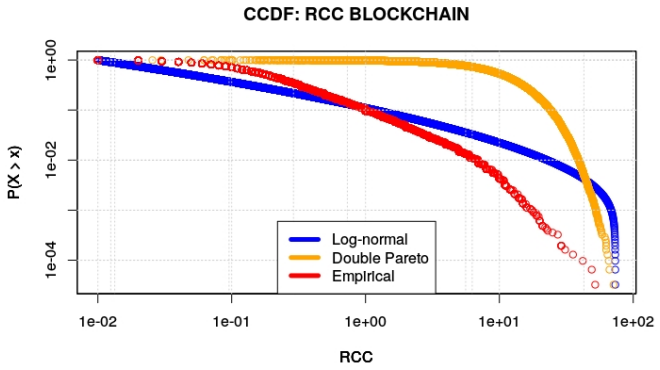

Fig. 5. RCC Blockchain-oriented Software.

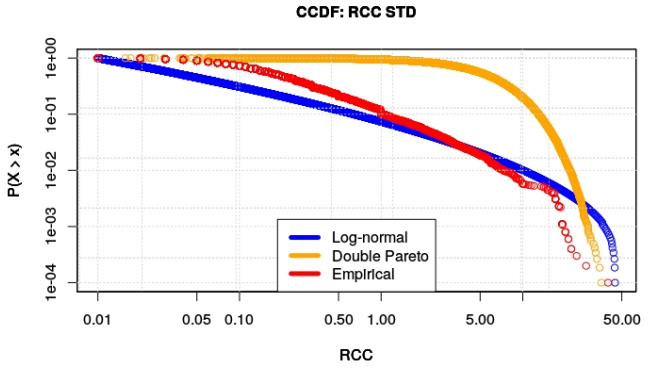

Fig. 6. RCC Traditional Software

Unlike the RCC metric, the differentiation between the two types of system is only possible with the initial portion of the double Pareto distribution. The statistical distributions appear different in the bulk of the analysed data, for classes having RCC values below the first regime threshold, which is around 20 for Blockchain and around 10 for Traditional, while they are similar along the queue of the distributions.

These results show that the metric Ration Comment to Code is the most different in Blockchain and Traditional projects and they can be easily distinguished using both best fitting distributions. In both cases, the fit is reasonably good for lower values in Blockchain software and higher value in Traditional software.

The results show that systems developed for Blockchain and Traditional software present different metrics distributions and statistical distributions providing fits for the metrics for all the Blockchain and Traditional projects have been identified. The best fitting parameters have been successfully used for a comparison of the metric distributions.

The double Pareto distribution fit for NOS metric reveals that major differences can be seen along the queue of the distributions. The same analysis with RCC metric, reveals that only the initial portion of the distribution shows differences between the two sets of projects. The dispersion parameter associated with the log-normal distribution fit for RCC metric can additionally be used for distinguishing the two sets.

V. THREATS TO VALIDITY

We considered Traditional software projects with different domains whilst we considered blockchain project oriented toward a specific domain. Although this may be a bias, it is eased considering that the blockchain ecosystem consists of several areas of interest such as distributed computing, web application, networking etc.

VI. CONCLUSION

We studied ten systems (five general purpose software and five blockchain oriented systems), in order to examine similarities and differences among Traditional and Blockchain-oriented projects by mean of the analysis of metrics statistical distributions. We found that log-normal and double Pareto distributions are sound for best fitting three metrics of every system analysed; this allowed a comparison among the two chosen sets. We found that it is possible to distinguish the two sets using the Average Cyclomatic (AC), Ration Comment To Code (RCC) and Number Of Statements (NOS). Among them, the Average Cyclomatic (AC), shows marked asymmetry for blockchain software in the form parameter, meaning that the Average Cyclomatic for blockchain projects presents a higher number of outliers in the top part of the metric distribution.

Our study showed how metrics analysis could successfully identify differences in programming practices, in particular, related to the use of Cyclomatic Complexity and the Number of Comments per Line of Code in Blockchain and Traditional software, thus revealing meaningful differences between the two projects domain.

REFERENCES

[1] M. Alshayeb and W. Li. An empirical validation of object-oriented metrics in two different iterative software processes. *Software Engineering, IEEE Transactions on*, 29(11):1043–1049, 2003.

[2] G. Baxter, M. Frean, J. Noble, M. Rickerby, H. Smith, M. Visser, H. Melton, and E. Tempero. Understanding the shape of java software. *SIGPLAN Not.*, 41(10):397–412, Oct. 2006.

[3] S. Bragagnolo, H. Rocha, M. Denker, and S. Ducasse. Smartinspect: solidity smart contract inspector. In *2018 International Workshop on Blockchain Oriented Software Engineering (IWBOSE)*, pages 9–18. IEEE, 2018.

[4] S. R. Chidamber, D. P. Darcy, and C. F. Kemerer. Managerial use of metrics for object-oriented software: An exploratory analysis. *Software Engineering, IEEE Transactions on*, 24(8):629–639, 1998.

[5] S. R. Chidamber and C. F. Kemerer. A metrics suite for object oriented design. *IEEE Transactions on Software Engineering*, 20(6):476–493, Jun 1994.

[6] S. R. Chidamber and C. F. Kemerer. A metrics suite for object oriented design. *Software Engineering, IEEE Transactions on*, 20(6):476–493, 1994.

[7] T. DeMarco. *Controlling software projects : management, measurement and estimation*. Yourdon Press, New York, NY, 1982.

[8] G. Destefanis, S. Counsell, G. Concas, and R. Tonelli. Software metrics in agile software: An empirical study. In *Agile Processes in Software Engineering and Extreme Programming*, pages 157–170. Springer, 2014.

[9] G. Destefanis, M. Marchesi, M. Ortu, R. Tonelli, A. Bracciali, and R. Hierons. Smart contracts vulnerabilities: a call for blockchain software engineering? In *Blockchain Oriented Software Engineering (IWBOSE), 2018 International Workshop on*, pages 19–25. IEEE, 2018.

[10] G. Destefanis, M. Ortu, S. Porru, S. Swift, and M. Marchesi. A statistical comparison of java and python software metric properties. In *Proceedings of the 7th International Workshop on Emerging Trends in Software Metrics*, pages 22–28. ACM, 2016.

[11] G. Destefanis, S. Qaderi, D. Bowes, J. Petrić, and M. Ortu. A longitudinal study of anti micro patterns in 113 versions of tomcat. In *Proceedings of the 14th International Conference on Predictive Models and Data Analytics in Software Engineering*, pages 90–93. ACM, 2018.

[12] G. Destefanis, R. Tonelli, E. Tempero, G. Concas, and M. Marchesi. Micro pattern fault-proneness. In *Software Engineering and Advanced Applications (SEAA), 2012 38th EUROMICRO Conference on*, pages 302–306. IEEE, 2012.

[13] G. Fenu, L. Marchesi, M. Marchesi, and R. Tonelli. The ico phenomenon and its relationships with ethereum smart contract environment. In *Blockchain Oriented Software Engineering (IWBOSE), 2018 International Workshop on*, pages 26–32. IEEE, 2018.

[14] S. Ibba, A. Pinna, M. Lunesu, M. Marchesi, and R. Tonelli. Initial coin offerings and agile practices. *Future Internet*, 10(11):103, 2018.

[15] P. Louridas, D. Spinellis, and V. Vlachos. Power laws in software. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 18(1):2, 2008.

[16] M. Marchesi, L. Marchesi, and R. Tonelli. An agile software engineering method to design blockchain applications. In *Proceedings of the 14th Central and Eastern European Software Engineering Conference Russia*, page 3. ACM, 2018.

[17] M. Mitzenmacher. Dynamic models for file sizes and double pareto distributions. *Internet Mathematics*, 1(3):305–333, 2004.

[18] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system, http://bitcoin.org/bitcoin.pdf.

[19] M. Orrú, E. Tempero, M. Marchesi, R. Tonelli, and G. Destefanis. A curated benchmark collection of python systems for empirical studies on software engineering. In *Proceedings of the 11th International Conference on Predictive Models and Data Analytics in Software Engineering*, page 2. ACM, 2015.

[20] M. Ortu, G. Destefanis, M. Orrú, R. Tonelli, and M. L. Marchesi. Could micro patterns be used as software stability indicator? In *Patterns Promotion and Anti-patterns Prevention (PPAP), 2015 IEEE 2nd Workshop on*, pages 11–12. IEEE, 2015.

[21] A. Pinna, R. Tonelli, M. Orrú, and M. Marchesi. A petri nets model for blockchain analysis. *The Computer Journal*, 61(9):1374–1388, 2018.

[22] S. Porru, A. Pinna, M. Marchesi, and R. Tonelli. Blockchain-oriented software engineering: Challenges and new directions. In *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*, pages 169–171, May 2017.

[23] A. Potanin, J. Noble, M. Frean, and R. Biddle. Scale-free geometry in oo programs. *Communications of the ACM*, 48(5):99–103, 2005.

[24] R. Shatnawi and Q. Althebyan. An empirical study of the effect of power law distribution on the interpretation of oo metrics. *ISRN Software Engineering*, 2013, 2013.

[25] C. P. Stark and N. Hovius. The characterization of landslide size distributions. *Geophysical Research Letters*, 28(6):1091–1094, 2001.

[26] R. Subramanyam and M. S. Krishnan. Empirical analysis of ck metrics for object-oriented design complexity: Implications for software defects. *Software Engineering, IEEE Transactions on*, 29(4):297–310, 2003.

[27] R. Tonelli, G. Destefanis, M. Marchesi, and M. Ortu. Smart contracts software metrics: a first study. *arXiv preprint arXiv:1802.01517*, 2018.

[28] M. Wohrer and U. Zdun. Smart contracts: security patterns in the ethereum ecosystem and solidity. In *Blockchain Oriented Software Engineering (IWBOSE), 2018 International Workshop on*, pages 2–8. IEEE, 2018.

[29] T. Zimmermann and N. Nagappan. Predicting defects using network analysis on dependency graphs. In *Proceedings of the 30th international conference on Software engineering*, pages 531–540. ACM, 2008.