A Preliminary Analysis of Software Metrics in Decentralised Applications

G. Ibba[1,3], S. Khullar[3], E. Tesfai[3], R. Neykova[3], S. Aufiero[2], M. Ortu[1], S. Bartolucci[2], G. Destefanis[3]
[1]University of Cagliari, Italy
[2]University College London, UK
[3]Brunel University London, UK
{giacomo.ibba,marco.ortu}@unica.it
{sabrina.aufiero.22,s.bartolucci}@ucl.ac.uk
{shivank.khullar,elaina.tesfai,rumyana.neykova,giuseppe.destefanis}@brunel.ac.uk

ABSTRACT

This study examines software metrics in decentralized applications (dApps) to analyze their structural and behavioral characteristics as they grow in complexity. Sixty dApps were categorized into Small (3 to 29 contracts), Medium (30 to 46 contracts), and Large (47 to 206 contracts) based on their contract count. Initial analysis showed a non-normal data distribution, leading to the use of Spearman's correlation method. Findings revealed that Medium dApps have strong correlations between metrics like 'Average Local Variables' and 'Maximum Local Variables', while Large dApps show higher correlations between 'Number of Functions' and 'State Variable Count', indicating more complex contract structures. The higher Coupling Between Objects (CBO) in large dApps suggests increased interactions with other contracts or libraries, potentially elevating security risks. These insights are valuable for developers and stakeholders in the blockchain and IoT sectors, aiding in understanding how dApps evolve with increasing complexity and the implications on software metric relationships.

CCS CONCEPTS

• **Do Not Use This Code → Generate the Correct Terms for Your Paper**; *Generate the Correct Terms for Your Paper*; Generate the Correct Terms for Your Paper; Generate the Correct Terms for Your Paper.

KEYWORDS

Do, Not, Us, This, Code, Put, the, Correct, Terms, for, Your, Paper

ACM Reference Format:
G. Ibba[1,3], S. Khullar[3], E. Tesfai[3], R. Neykova[3], S. Aufiero[2], M. Ortu[1], S. Bartolucci[2], G. Destefanis[3]. 2023. A Preliminary Analysis of Software Metrics in Decentralised Applications. In *Proceedings of the Fifth ACM International Workshop on Blockchain-enabled Networked Sensor Systems (BlockSys '23).* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3628354.3629533

BlockSys '23, November 12, 2023, Istanbul, Turkiye

ACM ISBN 978-8-4007-0439-0/23/11.
https://doi.org/10.1145/3628354.3629533.

1 INTRODUCTION

Blockchain technology introduces new possibilities for improving trust and privacy in networked sensor systems across different areas. Decentralised applications (dApps) play a key role in this, providing a solid platform for developing solutions that effectively manage sensor data while ensuring data protection and incentivized sharing. At the core of dApps are smart contracts—self-executing contracts with the terms directly written into code, enabling automated transactions on the blockchain. The growing number of dApps, especially within the Internet of Things (IoT) and smart city domains, emphasizes the importance of understanding their software metrics to improve development, ensure security, and enhance performance.

This study provides an initial analysis of the software metrics of dApps, aiming to explore how these metrics vary with the size and complexity of the dApps. By categorizing 60 dApps (selected from the DAppScan repository[1]) [12] into three distinct groups based on their contract count—Small, Medium, and Large, we aim to analyze the distribution and correlation of software metrics at both contract and function levels. Our analysis examines whether and how the distributions of these metrics, and the correlations among them, change with the dApp size. This preliminary analysis contributes to the understanding around dApps' structure and behavior, setting the stage for more in-depth future studies.

2 RELATED WORK

Several studies have previously explored software metrics within traditional applications to understand their structure and behavior better[4, 5]. Focusing on object-oriented (OO) software, one of the pioneering efforts to address this concern is credited to Chidamber and Kemerer (CK), who proposed the widely recognized CK metrics suite for OO software systems [1]. Numerous empirical studies have since underscored significant correlations between certain CK metrics and bug-proneness [3, 7, 10]. Metrics defined on software graphs have also been explored, with findings correlating them to software quality [13]. Transitioning to the blockchain domain, in a recent work, Ibba et al. [8] developed a tool for employing complex Networks Analysis on dApps[2], in order to help with the identification of vulnerability and code optimisation.

Ortu et al. [9] compared Blockchain-Oriented Software (BOS) and traditional software using 10 metrics, finding significant differences

[1]https://github.com/InPlusLab/DAppSCAN/tree/main

in the distribution of Average Cyclomatic and Ratio Comment To Code metrics, and the Number of Statements metric.

Tonelli et al. [11] analyzed 85,000 Smart Contracts on the Ethereum blockchain to understand how their constraints are reflected in specific software metrics compared to traditional software. Findings showed that while Smart Contracts exhibit more restricted metric ranges, their lines of code follow a log-normal distribution akin to traditional software, hinting at some shared characteristics despite the unique constraints of blockchain environments.

Our study extends the investigation to decentralized applications on the Ethereum blockchain, categorizing them based on their contract count to explore the distribution and correlation of software metrics across different complexity levels. This analysis not only sheds light on how dApp size and complexity interact with various software metrics but also sets a foundation for future in-depth studies aimed at understanding dApps' evolution and potential optimization strategies.

3 METHODOLOGY

The metrics for this analysis were collected through a combination of our in-house analysis tools and Slither[6], a well-known static analysis framework for smart contracts. In analyzing the software metrics of decentralized applications (dApps), it is fundamental to select an appropriate method that accurately reflects the underlying relationships among the metrics. An initial assessment of the data revealed a non-normal distribution, which is a common occurrence in real-world data, especially in a relatively new and rapidly evolving domain like blockchain. Traditional correlation methods such as Pearson's correlation assume a linear relationship and a normal distribution of data, which could lead to misleading conclusions in our case.

Spearman's correlation, on the other hand, does not make any assumptions about the distribution of the data and measures the strength and direction of the monotonic relationship between variables. It evaluates the rank-order relationship between two variables, making it a more robust choice for this analysis. This method is well-suited for our dataset, allowing for a more accurate exploration of correlations between software metrics across different sizes and complexities of dApps.

We proceeded to categorize the dApps into distinct groups based on their structural complexity, as represented by the number of contracts they contain. This categorization is fundamental to our analysis as it provides a systematic approach to understanding how the structural and behavioral characteristics of dApps vary with size and complexity, laying a solid foundation for the subsequent analysis.

The categorization of dApps into Small, Medium, and Large groups based on the number of contracts they contain is a practical approach derived from the characteristics of our dataset. The specific ranges (3 to 29, 30 to 46, 47 to 206) for these categories were selected to create a balanced division that allows for meaningful comparison and analysis across groups. However, this division is arbitrary and represents a significant limitation of the study. It is dictated by the distribution and the nature of the dApps available in our dataset rather than a universally accepted standard. This categorization helped in understanding the variation in software metrics among dApps of different sizes and complexities in the context of our dataset, though the defined ranges may not hold or be relevant for a different dataset or in a broader context. Future studies may benefit from a more standardized or universally accepted method of categorization, or by exploring alternative methods that might provide a more nuanced understanding of dApp complexity and size.

We categorized these applications based on their structural complexity, specifically focusing on the number of contracts they contain. This approach allows us to capture the nuanced differences in DApps, which can range from simple prototypes to highly complex ecosystems.

- Small DApps: This category includes DApps with a number of contracts ranging from 3 to 29. They are often simpler, either being in the prototype stage or targeting very specific use-cases. For example, the DApp "Async" has just one file, three contracts, and eight functions.
- Medium DApps: DApps falling into this category have a number of contracts ranging from 30 to 46. These DApps are more complex than those categorized as "Small" but not as intricate as the "Large" DApps. They often address broader use-cases and incorporate more complex functionalities. An example would be "AliumSwap" with 24 files, 30 contracts, and 240 functions.
- Large DApps: These are highly complex DApps that contain a number of contracts ranging from 47 to 206. They often serve diverse functions and may be part of a larger blockchain ecosystem. For instance, "Loopring" has 200 files, 206 contracts, and 1591 functions.

The primary metric driving this categorization is the Number of Contracts. It offers a quantitative measure of a DApp's complexity and potentially its functional diversity. By organizing the DApps according to these categories, this study aims for a systematic and structured approach to understanding how size and complexity relate to other structural and security metrics. This approach helps identify specific trends or patterns that may be unique to DApps of certain sizes, thereby adding depth and granularity to the study's findings.

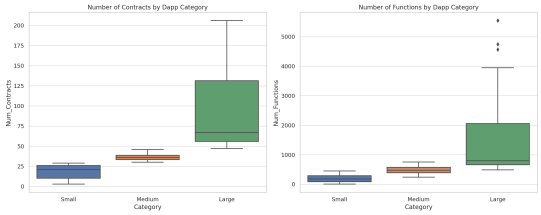

Figure 1: Boxplots - number of contract and functions

Figure 1 presents the box plots showing the distributions of the number of contracts and functions for each category (Small, Medium, Large):

The first plot shows the distribution of the number of contracts. The second plot shows the distribution of the number of functions. In both plots, the central line in each box indicates the median of the

data, while the top and bottom edges of the box show the interquartile range. The "whiskers" extend to 1.5 times the interquartile range, and any data points beyond that are considered outliers.

Before studying contract and function level metrics, it is fundamental to understand the distribution type for each metric.

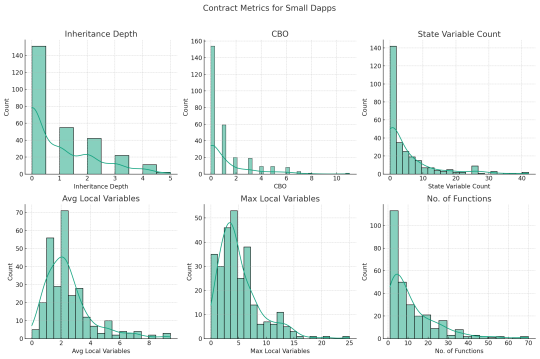

Figure 2: Contract Metrics for Small DApp

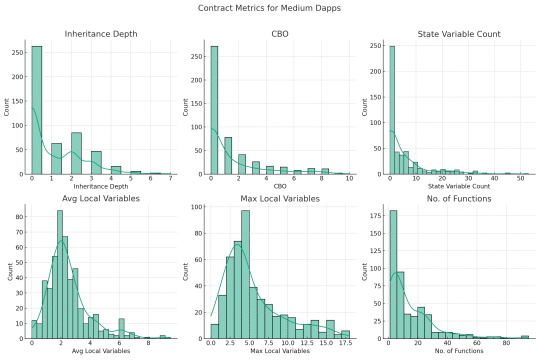

Figure 3: Contract Metrics for Medium DApp

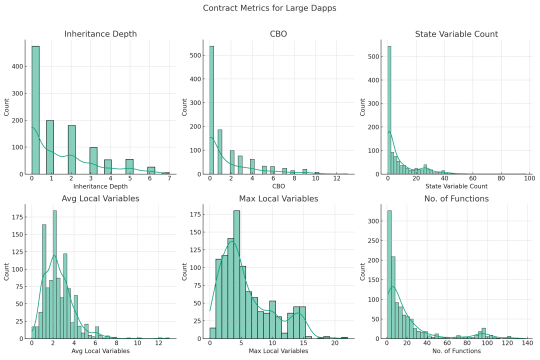

Figure 4: Contract Metrics for Large DApp

To assess the distribution type, histograms were plotted for each metric, both at the contract and function levels. These histograms were generated separately for Small (Fig. 2), Medium (Fig. 3), and Large dApps (Fig. 4) to observe any category-specific trends. Kernel density estimates were also plotted to provide a smooth, continuous representation of the data distribution. The histograms revealed that most metrics, irrespective of dApp category, exhibited a right-skewed distribution. Based on this observation, non-parametric correlation measures like Spearman's were selected for the subsequent correlation analysis.

To prepare for correlation analysis, the normality of metric distributions was assessed using both Shapiro-Wilk and Kolmogorov-Smirnov tests, confirming the non-normal nature of the data. This led to the selection of non-parametric correlation methods, namely Kendall's and Spearman's, to analyze the relationships between different metrics. The findings from these correlation analyses offered insights into the interplay between various contract and function attributes, providing a nuanced understanding of dApp characteristics across different sizes and complexities.

Table 1: Summary of Small Dapps

Dapp_Name	Num_Files	Num_Contracts	Num_Functions
Async	1	3	8
Gifto	6	3	55
Polymath	6	6	35
BitcoinSB_V2	3	7	81
CGU	8	8	84
Codex_Altash	3	9	84
Holdefi	7	14	157
1inch	8	15	84
StackerVC	9	17	203
NZ-Beam	4	18	222
PikaPerpv2	8	20	176
XSwap	16	22	188
Saddle	12	22	259
GHST	22	22	143
Donut	15	23	167
ImpossibleSwap	23	26	322
Dodo	26	26	314
BackstopSyndicate	15	28	299
BCUBE	9	29	448
Crodex	18	29	249
FarmHero	6	29	336
Avatar	17	29	336

4 RESULTS

The following **contract-related** metrics were considered for this analysis:

- **Inheritance Depth**: Measures how many layers of inheritance a contract has. A higher depth could indicate a more complex contract structure. Most Small and Medium dApps tend to have a lower inheritance depth compared to Large dApps, which often employ multiple layers of inheritance for added functionality and modularity.
- **CBO (Coupling Between Objects)**: Indicates the number of other contracts or libraries that a contract interacts with. Higher coupling may lead to increased complexity and potential risks. The Coupling Between Objects (Contracts in our

Table 2: Summary of Medium Dapps

Dapp_Name	Num_Files	Num_Contracts	Num_Functions
AliumSwap	24	30	240
SushiSwap	20	31	322
IronLend	26	31	733
Polynetwork	31	32	385
IDLEGovernance	26	33	387
GoodGhosting	26	34	388
Coordinape	14	35	416
IDLEFinance	23	35	478
Gods_Unchained	25	36	574
MeritCircle	13	36	672
Axie_Infinity	25	36	574
TokenCard	24	38	557
ShibaNova	29	38	469
UMA	32	39	310
LuckyChip	27	43	633
Qubit	35	45	754
OriginDollar	30	46	405
COGI	8	46	556

Table 3: Summary of Large Dapps

Dapp_Name	Num_Files	Num_Contracts	Num_Functions
MarbleCards	25	47	488
Amplify	42	49	815
DarkCrypto	39	49	776
DForce	42	54	746
Tanchessv	37	56	665
POA-DPOS	42	56	1124
MetaVaultV2	43	57	609
NaosFormation	49	61	739
AAVE3	59	62	516
ICHI	55	65	986
GammaProtocol	67	69	625
88mph	50	74	765
Atlantis	49	76	647
DSG	51	85	1157
CREAMFinanceFlashloan	71	130	4744
Rikkei	76	135	3444
Compound	85	140	4567
CREAMFinanceCompound	78	144	5544
Venus	89	150	3949
Loopring	200	206	1591

case) is generally higher in Large dApps, suggesting more interactions with other contracts or libraries. Small dApps tend to have lower coupling, indicating simpler architectures.

- **State Variable Count**: Represents the number of state variables in a contract. A higher count could lead to more complex contract interactions. Large dApps generally employ more state variables, likely to manage more complex states and operations. Small and Medium dApps usually have fewer state variables, reflecting simpler logic and state management.
- **Avg Local Variables**: The average number of local variables used across all functions in a contract. This can be an indicator of how much temporary storage a contract uses. Across all categories, the average number of local variables tends to be moderate, indicating a balance in the use of temporary storage for function computations.
- **Max Local Variables**: The maximum number of local variables used in any single function within a contract. The metrics show occasional spikes in the number of maximum local variables in functions, especially in Large dApps, suggesting some functions may be doing more complex computations.
- **No. of Functions**: The total number of functions in a contract. This metric gives an idea of the contract's functionality and complexity. Observations Large dApps clearly have a higher number of functions, providing more services or features. Small dApps, in contrast, are simpler and offer fewer functionalities.

4.1 Function level metrics

The following discuss the Analysis of Function-Level Metrics Across Dapp Categories to further investigate the complexities of decentralized applications (dApps), the study also focuses on function-level metrics across Small, Medium, and Large dApps. These metrics offer a granular look into how individual functions within smart contracts are designed and implemented.

The function-level metrics analyzed are:

- **No. of Parameters**: Indicates the number of parameters a function takes. A higher number could make the function more complex and harder to use. Functions in Small dApps tend to have fewer parameters, implying simpler interfaces. In contrast, Medium and Large dApps often have functions with more parameters, allowing for more complex interactions.
- **Nesting Depth**: Represents the depth of nested loops and conditionals within a function. Deeper nesting can make a function harder to understand and maintain. Higher nesting depths are more frequently observed in Large and Medium dApps, suggesting more intricate logic and conditions. Small dApps generally have functions with lower nesting depths.
- **Function Calls**: Counts the number of times a function calls other functions. Frequent calls can lead to intricate function behaviors and interactions. Functions in Large dApps usually make more calls to other functions, indicating a higher degree of modularity and potential complexity. This is less common in Small and Medium dApps.
- **Cyclomatic Complexity**: Measures the number of linearly independent paths through a function's source code. Higher values denote more complex functions. This metric tends to be higher in functions belonging to Large dApps, pointing to more complicated control flow. Functions in Small and Medium dApps usually have lower cyclomatic complexity, implying simpler logic.
- **Local Variable Count**: Indicates the number of local variables within a function. A higher count could imply more complex computations and logic within the function. Large dApps typically have functions with more local variables, likely due to more complex calculations or data manipulations. The count is generally lower in Small and Medium dApps.

4.2 Evaluating the Metrics Distribution

Before proceeding with the correlation analysis of contract and function-level metrics across different categories of decentralized applications, it is crucial to understand the distribution type for each metric. This initial step is important as the type of distribution can significantly influence the choice of correlation measure used. For instance, Pearson's correlation is most effective when the data is normally distributed, but may produce misleading results if the data is skewed or contains outliers. On the other hand, non-parametric measures like Spearman's and Kendall's correlation are more robust against such irregularities.

The Shapiro-Wilk test, a widely-accepted statistical test for normality, was employed on each of the contract and function-level metrics, segregated by the dApp categories: Small, Medium, and Large. The test outputs a p-value, where a value less than 0.05 typically suggests that the data does not follow a normal distribution. The p-values for all metrics across all categories were significantly less than 0.05, with function-level metrics even yielding a p-value of zero. These findings indicate that the metric distributions are not normal, thereby making a compelling case for the use of non-parametric correlation measures like Spearman's or Kendall's for the analysis.

As an additional layer of robustness to the normality assessment, the Kolmogorov-Smirnov (KS) test was executed on each of the contract and function-level metrics, categorized by the size of the dApps: Small, Medium, and Large. The KS test compares the empirical distribution function of the sample data with the cumulative distribution function of a specified theoretical distribution—in this case, the normal distribution. The p-values obtained for all metrics across each category were essentially zero, thereby rejecting the null hypothesis of normal distribution conclusively. These findings corroborate the results from the earlier Shapiro-Wilk test, reinforcing the decision to employ non-parametric correlation measures for the subsequent correlation analysis.

4.3 Spearman's Correlations

In examining the Spearman's Correlation matrices for contract metrics across small (Fig. 5), medium (Fig. 6), and large Dapps (Fig. 7), certain patterns emerge. For small Dapps, there are notable strong correlations between 'Inheritance Depth', 'State Variable Count', and 'Number of Functions' with values greater than 0.7. In contrast, medium Dapps display slightly diversified strong correlations, between 'Inheritance Depth' and 'State Variable Count' approaching 0.8, with 'Number of Functions' also being significantly related to these metrics. However, large Dapps exhibit a dilution in the strength of these correlations, with the strongest link being between 'State Variable Count' and 'Inheritance Depth' at around 0.78. It's interesting to note that as the Dapps grow in complexity (from small to large), the correlations between 'Avg Local Variables' and other metrics become more dispersed, suggesting that larger Dapps might have a broader variance in their contract structures. This comparative analysis provides an insight into how contract interactions and structures evolve with the size and complexity of Dapps.

In the evaluation of Spearman's correlation among function metrics within decentralized applications (dApps) of varying sizes,

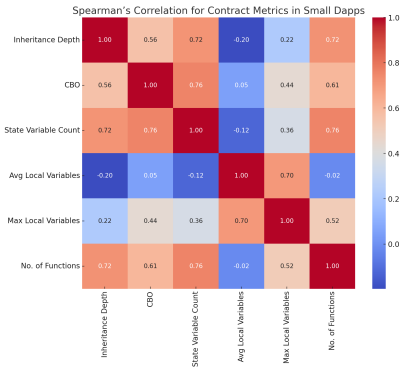

Figure 5: Correlation Metrics for Small DApp

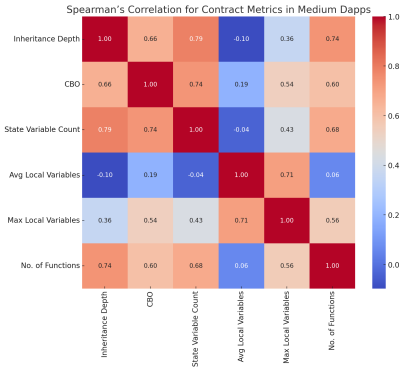

Figure 6: Correlation Metrics for Medium DApp

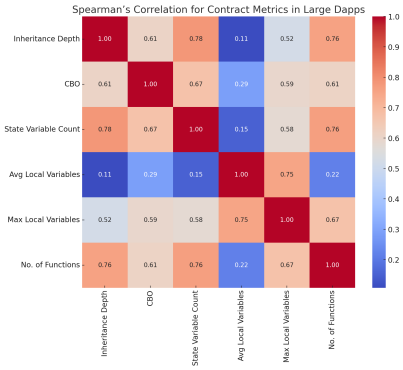

Figure 7: Correlation Metrics for Large DApp

distinct patterns emerge. For small dApps, there's a strong positive correlation between the number of parameters and the local variable count (0.83), suggesting that as functions increase their parameter count, they also tend to have more local variables. Medium-sized dApps show a similar trend, albeit slightly weaker (0.85). In large dApps, this correlation remains significant but decreases to 0.81. Interestingly, cyclomatic complexity exhibits a negative correlation with function calls for all dApp sizes: -0.38 for small, -0.39 for medium, and -0.42 for large. This result should be investigated

further. A negative correlation between cyclomatic complexity and function calls might initially seem counterintuitive because one might expect more complex functions to have more function calls. However, this negative correlation could be related to decomposition, e.g., developers may be breaking down complex logic into smaller, more manageable functions. This would mean fewer function calls within each complex function, as the logic is spread out. Higher cyclomatic complexity often involves more branching (if, else, switch, etc.). It is possible that in more complex functions, the logic is handled through conditional structures rather than function calls. It may reflect a particular design philosophy or best practice that advises against making multiple function calls within complex functions to make the code easier to understand and maintain.

It is also noteworthy that the correlation between the number of parameters and function calls is fairly consistent across dApp sizes, ranging from 0.22 to 0.24. Overall, these findings provide insights into the evolution of function design patterns as dApps scale.

The findings from our analysis carry implications for the development, security, and performance optimization of decentralized applications (dApps).

- Development Complexity: Our analysis reveals a clear correlation between the size of dApps and certain software metrics, which reflects an increase in development complexity as dApps scale. Understanding these correlations can help developers anticipate the challenges they may face as their dApps grow, enabling better planning and resource allocation.
- Security Considerations: The higher Coupling Between Objects (CBO) in large dApps suggests more interactions with other contracts or libraries, which could potentially introduce security risks. Moreover, the increased inheritance depth in larger dApps might also lead to a more complex contract structure, requiring more rigorous security auditing and testing to ensure robustness against potential threats.
- Performance Optimization: The analysis of function-level metrics provides insights into how individual functions within smart contracts are designed and implemented across different dApp sizes. The correlation between the number of parameters and the local variable count, for instance, could have implications for the performance and gas costs in Ethereum-based dApps. Understanding these patterns can help developers optimize their code to ensure efficient resource utilization, especially in larger dApps with more complex structures.
- Modular Design: The higher frequency of function calls in large dApps indicates a higher degree of modularity, which is essential for managing complexity in software development. This modularity might aid in isolating issues, enhancing maintainability, and promoting reusable code.

The analysis undertaken in this study aligns with an exploratory approach, aimed at uncovering initial insights and trends concerning the software metrics of decentralized applications (dApps) across varying sizes and complexities.

5 THREATS TO VALIDITY

The validity of this study is subject to several threats that need acknowledgment. The sample size of 60 dApps selected from the DAppScan repository is not representative of the broader spectrum of dApps, potentially introducing a selection bias if the repository lacks diversity or has a specific focus. The categorization of dApps into Small, Medium, and Large based on the number of contracts is somewhat arbitrary and derived from the dataset on hand. This categorization may not capture the true essence of complexity and size, thereby potentially oversimplifying the heterogeneity of dApps. The choice of metrics for analysis, although based on software engineering principles, may not encompass all relevant aspects of dApp complexity and functionality, and there might be other metrics not considered that could provide additional or alternative insights. The data exhibited a non-normal distribution, which led to the use of non-parametric correlation measures like Spearman's correlation. While these measures are robust against certain irregularities, they may not capture all relationships or nuances present in the data. The findings, may have limited generalisability beyond the specific set of dApps analyzed, and the rapid evolution of blockchain technology and dApp development practices may impact the relevance and applicability of the findings over time. As an exploratory study, the analysis aims to present initial insights and trends rather than confirm predefined hypotheses. The findings should be interpreted as preliminary, necessitating further confirmatory analyses to establish stronger causal or correlational relationships. Lastly, the accuracy and precision of the tools and methods used to collect and analyze the metrics can also pose a threat to validity. Any inconsistencies or errors in measurement could potentially affect the reliability and reproducibility of the findings. Through acknowledging these threats to validity, we aim to provide a transparent account of the limitations inherent in our study and lay the groundwork for further research that can build upon, validate, or refine the preliminary results presented.

6 CONCLUSIONS

This preliminary study analysed software metrics of dApps on the Ethereum blockchain, categorizing them into Small, Medium, and Large based on contract count. The analysis showed that as dApps scale, certain metrics such as Inheritance Depth and Coupling Between Objects (CBO) increase, indicating more complex contract structures and interactions with other contracts or libraries. Larger dApps not only have more contracts but also more complex contracts, which could potentially introduce security risks.

On the function level, a consistent relationship was found between the number of parameters and local variable count across all dApp sizes. Additionally, a negative correlation between cyclomatic complexity and function calls was observed, suggesting a possible trend of decomposing complex logic into smaller, more manageable functions in larger dApps.

These findings are important for developers and stakeholders in the blockchain and IoT sectors as they provide a clearer understanding of how dApps evolve with increasing complexity. This information can be valuable for better planning, security auditing, and performance optimization in dApp development. The results also provide a basis for more in-depth future studies on software metrics in decentralized applications.

7 ACKNOWLEDGMENT

S.B., G.D., R.N. and M.O. acknowledge support from the Ethereum foundation grant FY23-1048

REFERENCES

[1] 1994. A metrics suite for object oriented design. *IEEE Transactions on software engineering*, 20, 6, 476–493.

[2] Sabrina Aufiero, Giacomo Ibba, Silvia Bartolucci, Giuseppe Destefanis, Rumyana Neykova, and Marco Ortu. 2023. The network structure of smart contracts in ethereum dapps. *Complex Networks 2023 (to appear)*.

[3] Victor R Basili, Lionel C. Briand, and Walcélio L Melo. 1996. A validation of object-oriented design metrics as quality indicators. *IEEE Transactions on software engineering*, 22, 10, 751–761.

[4] Giulio Concas, Giuseppe Destefanis, Michele Marchesi, Marco Ortu, and Roberto Tonelli. 2013. Micro patterns in agile software. In *Agile Processes in Software Engineering and Extreme Programming: 14th International Conference, XP 2013, Vienna, Austria, June 3-7, 2013. Proceedings 14*. Springer, 210–222.

[5] Giuseppe Destefanis, Marco Ortu, Simone Porru, Stephen Swift, and Michele Marchesi. 2016. A statistical comparison of java and python software metric properties. In *Proceedings of the 7th International Workshop on Emerging Trends in Software Metrics*, 22–28.

[6] Josselin Feist, Gustavo Grieco, and Alex Groce. 2019. Slither: a static analysis framework for smart contracts. In *2019 IEEE/ACM 2nd International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB)*. IEEE, 8–15.

[7] Tibor Gyimóthy, Rudolf Ferenc, and Istvan Siket. 2005. Empirical validation of object-oriented metrics on open source software for fault prediction. *IEEE Transactions on Software engineering*, 31, 10, 897–910.

[8] Giacomo Ibba, Sabrina Aufiero, Silvia Bartolucci, Rumyana Neykova, Marco Ortu, Roberto Tonelli, and Giuseppe Destefanis. 2023. Mindthedapp: a toolchain for complex network-driven structural analysis of ethereum-based decentralised applications. *arXiv preprint arXiv:2310.02408*.

[9] Marco Ortu, Matteo Orrú, and Giuseppe Destefanis. 2019. On comparing software quality metrics of traditional vs blockchain-oriented software: an empirical study. In *2019 IEEE International Workshop on Blockchain Oriented Software Engineering (IWBOSE)*. IEEE, 32–37.

[10] Ramanath Subramanyam and Mayuram S. Krishnan. 2003. Empirical analysis of ck metrics for object-oriented design complexity: implications for software defects. *IEEE Transactions on software engineering*, 29, 4, 297–310.

[11] Roberto Tonelli, Giuseppe Antonio Pierro, Marco Ortu, and Giuseppe Destefanis. 2023. Smart contracts software metrics: a first study. *Plos one*, 18, 4, e0281043.

[12] Zibin Zheng, Jianzhong Su, Jiachi Chen, David Lo, Zhijie Zhong, and Mingxi Ye. 2023. Dappscan: building large-scale datasets for smart contract weaknesses in dapp projects. *arXiv preprint arXiv:2305.08456*.

[13] Thomas Zimmermann and Nachiappan Nagappan. 2008. Predicting defects using network analysis on dependency graphs. In *Proceedings of the 30th international conference on Software engineering*, 531–540.