Assignment 1 FINAL REPORT

Title: Analyzing the Effect of Size on Maintainability in Java Projects

By Bhavya Gunnapaneni

Spandana Bellamkonda

Section 1: Introduction

As a software developer, I have always been fascinated by the concept of maintainability in software systems. The ability to easily upgrade, rectify, adjust, and extend a software project is crucial for its long-term success. With this in mind, I embarked on a research project to explore the relationship between the size of Java projects and their maintainability. This assignment presents the culmination of my efforts and findings in the form of a final report. Java, being one of the most widely used programming languages across various domains, presents an ideal platform for conducting this research. Its versatility and prevalence in applications ranging from online platforms to mobile apps and corporate systems make it a suitable choice for investigating the impact of size on maintainability. By focusing on Java projects, the results of this study can have practical implications for a wide range of software development teams and projects.

To ensure a systematic and structured approach, I adopted the Goal-Question-Metric (GQM) framework for organizing the analysis. This framework allowed me to define clear research goals, generate relevant questions, and identify metrics that would provide meaningful answers. By employing the GQM methodology, I aimed to ensure that the conclusions drawn from the research would be both relevant and useful for developers and businesses seeking to enhance their software development processes. The primary objective of this study was to examine the relationship between the size of Java projects and their maintainability, development teams can make informed decisions during software system development and implementation, ultimately leading to the creation of more maintainable code.

A fundamental question driving this research is whether the number of lines of code (LoC) in a Java project significantly affects its maintainability. Exploring this question is crucial to identifying trends and patterns that can guide the development of best practices for overseeing and sustaining large-scale Java projects. Therefore, LoC serves as a vital metric in our investigation, enabling us to assess the impact of project size on maintainability accurately. In addition to LoC, we selected three complementary metrics to provide a comprehensive evaluation of maintainability in Java projects:

Coupling Between Objects (CBO) Response for a class (RFC), and Tight Class Cohesion (TCC). CBO measures the level of interdependency between classes, with lower coupling indicating higher maintainability. RFC counts the number of methods that can be executed in response to messages received by a class, with lower RFC values indicating better maintainability. TCC evaluates the cohesion of a class's methods, with higher values indicating stronger maintainability.

By analyzing these metrics alongside the size of Java projects, measured in LoC, I aim to establish a clear and definitive link between project size and maintainability. Through this research, I strive to provide insights that will empower development teams to make informed decisions, adopt best practices, and improve the overall maintainability of their Java projects. Throughout this report, I will present the methodology employed, the data collected, and the analysis performed to investigate the relationship between size and maintainability in Java projects. Additionally, I will discuss the implications of the findings and propose recommendations for software development teams and businesses. In conclusion, this research project delves into the crucial area of software maintainability by examining the effect of size on maintainability in Java projects. By employing the GQM framework and utilizing metrics such as LoC, CBO, RFC, and TCC, we aim to shed light on the relationship between project size and maintainability. The insights gained from this study have the potential to guide the development of best practices and strategies that enhance the maintainability of Java projects, benefiting developers and businesses alike.

Section 2: Dataset Description

Project Name	**Repository URL**	**Description**	**Size (LoC)**	**Developers**	**Age (Years)**
Netflix/Priam	https://github.com/Netflix/Priam	Co-Process for backup/reco	13291	44	12

		very, Token Management, and Centralized Configuration management for Cassandra.			
glowroot/glowroot	https://github.com/glowroot/glowroot	Easy to use, very low overhead, Java APM	32537	20	9
apache/rocketmq	https://github.com/apache/rocketmq	Mirror of Apache RocketMQ	14799	357	6
VazkiiMods/Botania	https://github.com/VazkiiMods/Botania	A magic themed tech mod for Minecraft based on nature and plant life.	90095	169	9
neo4j/neo4j	https://github.com/neo4j/neo4j	Graphs for Everyone	544557	221	11

Section 3: Tool Description

During the course of my research project, I employed the CKJM tool to gather essential measurements of Chidamber and Kemerer (C&K) metrics for the classes in the selected

Java projects. The decision to utilize CKJM was driven by several key considerations that make it an ideal tool for our analysis. CKJM is a dedicated software tool specifically designed to work with Java-based projects, enabling the computation of C&K metrics for object-oriented programming. These metrics, including Coupling Between Objects (CBO), Response for a Class (RFC), and Tight Class Cohesion (TCC), among others, are directly relevant to our research inquiry and provide valuable insights into the relationship between project size and maintainability in Java projects.

One of the major advantages of CKJM is its user-friendly command-line interface, which simplifies the installation and operation of the tool. With its straightforward setup process, it accepts Java bytecode files in the .class format and generates the desired metrics in a clear and concise manner. This simplicity empowers users to efficiently acquire the necessary measurements for analysis without grappling with the complexities of a more intricate tool. Moreover, CKJM is an open-source software tool, accompanied by a comprehensive user guide. This transparency in functionality and metric acquisition process ensures the reliability and trustworthiness of the metrics obtained for analytical purposes. Users can refer to the user guide to gain a deeper understanding of the tool's capabilities and the metrics it provides, further enhancing their confidence in the results.

Flexibility is another key feature of CKJM, as it allows users to choose the specific C&K metrics they want to compute. This flexibility proved invaluable for our study, as we focused on the CBO, RFC, and TCC metrics, which are fundamental in assessing maintainability in Java projects. By selecting the most relevant metrics, we could obtain a comprehensive understanding of the relationship between project size and maintainability. The widespread utilization and recognition of CKJM within the software engineering research sphere add to its credibility as a reliable tool. It has been extensively employed in numerous academic investigations, earning its reputation as a trusted instrument. The tool's established presence and the acknowledgment it has garnered provide further validation for the reliability of the derived metrics.

To access the CKJM software metric tool, interested users can visit the following Uniform Resource Locator: https://github.com/mauricioaniche/ck.

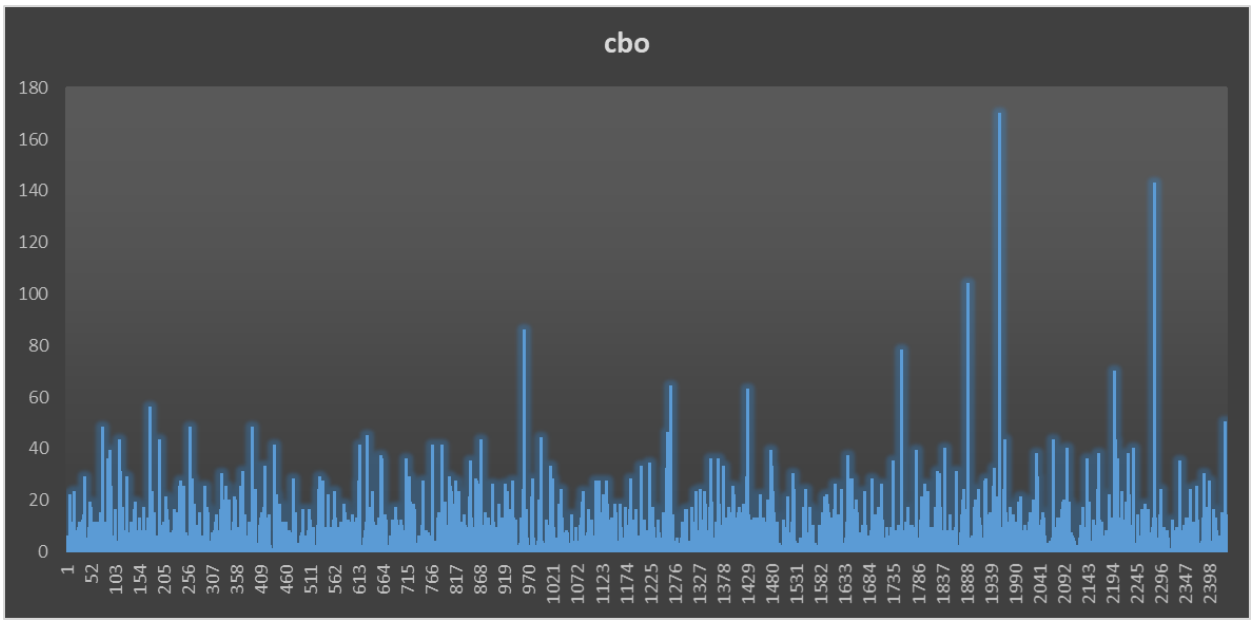
The above-mentioned website provides the necessary resources to download and install CKJM, ensuring easy accessibility for researchers and practitioners alike. In our study, the first step involved retrieving the source code of each selected project from the GitHub platform. Subsequently, the projects were compiled to generate the Java bytecode files (.class). We then utilized the CKJM tool to acquire the C&K metrics measurements for each class in the compiled projects.

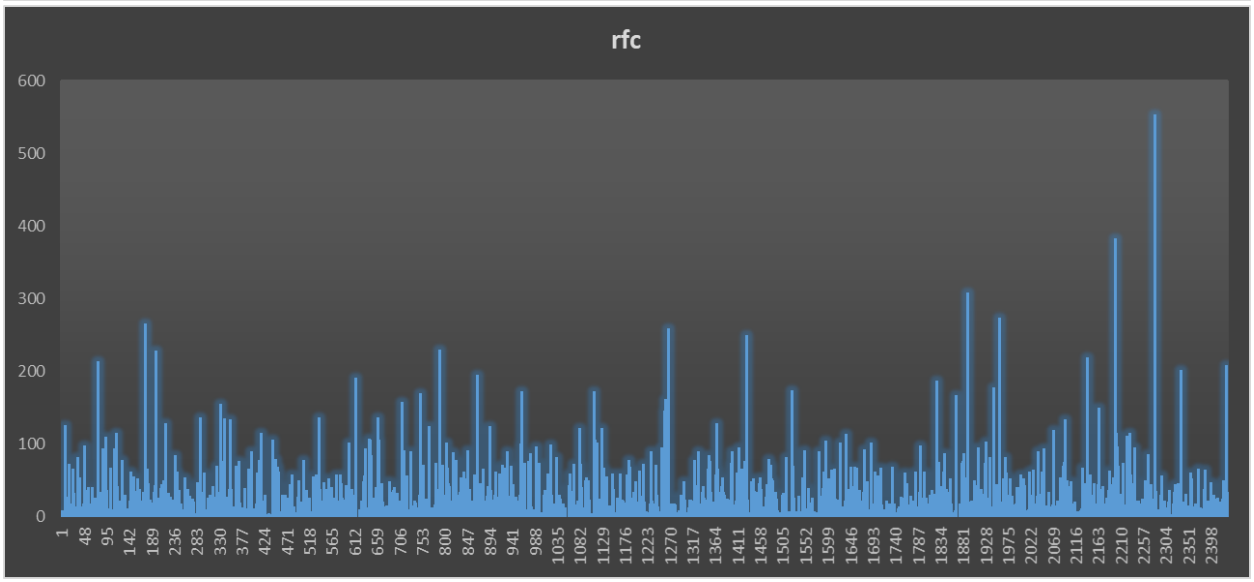
These metrics formed the basis of our examination of the correlation between project size and maintainability in the designated Java projects, as described in Section 4 of our report. In conclusion, the CKJM tool proved to be an invaluable asset in our research project, offering comprehensive metrics analysis for Java projects. Its Java-specific focus, user-friendly interface, open-source nature, flexibility in metric selection, and established reputation all contributed to its effectiveness and reliability. By leveraging CKJM, we were able to gather accurate and meaningful measurements that shed light on the relationship between project size and maintainability in Java projects, empowering software developers and teams with insights to enhance their development processes.

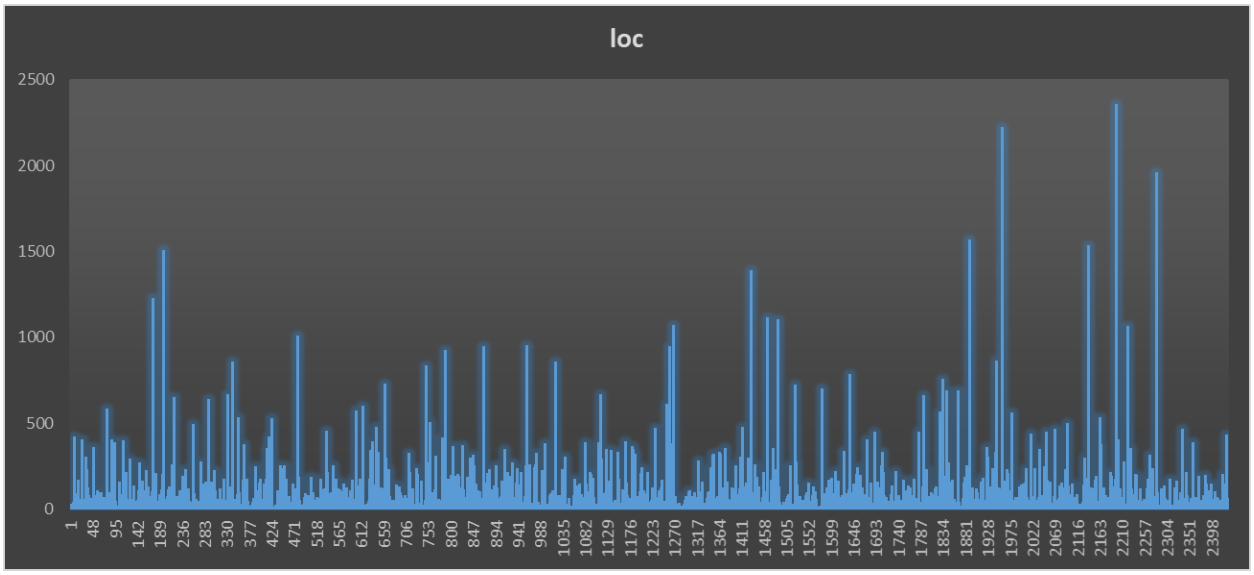
Section 4: Project Results

Project1: rocketmq

	CBO	**RFC**	**LOC**
Max	170	552	2354
Mode	1	0	5
Median	3	6	28
Std Deviation	9.640308	32.60403	151.0807
Average	6.526316	17.80633	72.63898

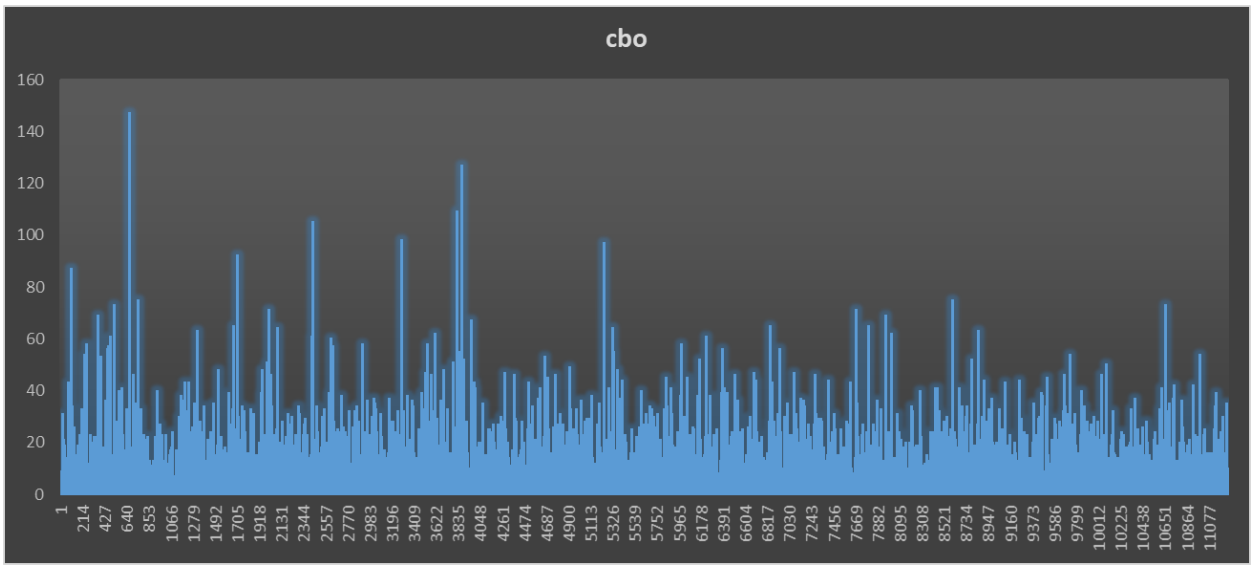

cbo
180
160
140
120
100
80
60
40
20
0
1
52
103
154
205
256
307
358
409
460
511
562
613
664
715
766
817
868
919
970
1021
1072
1123
1174
1225
1276
1327
1378
1429
1480
1531
1582
1633
1684
1735
1786
1837
1888
1939
1990
2041
2092
2143
2194
2245
2296
2347
2398

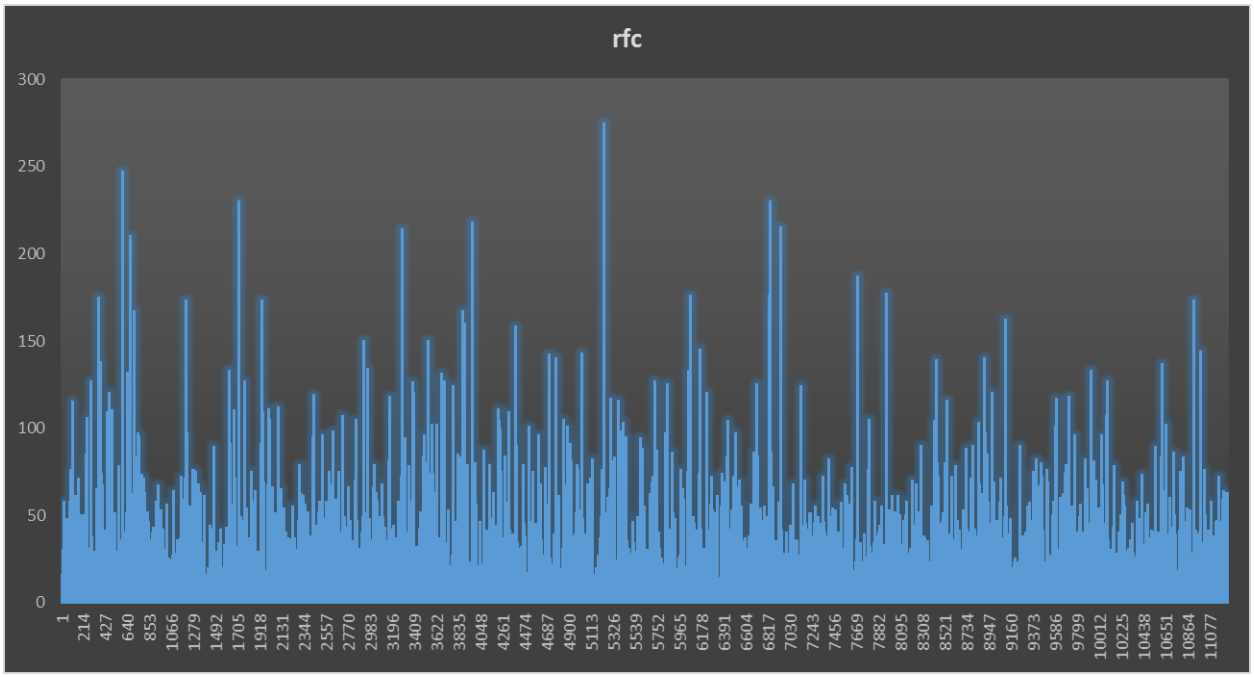

rfc
600
500
400
300
200
100
0
1
48
95
142
189
236
283
330
377
424
471
518
565
612
659
706
753
800
847
894
941
988
1035
1082
1129
1176
1223
1270
1317
1364
1411
1458
1505
1552
1599
1646
1693
1740
1787
1834
1881
1928
1975
2022
2069
2116
2163
2210
2257
2304
2351
2398

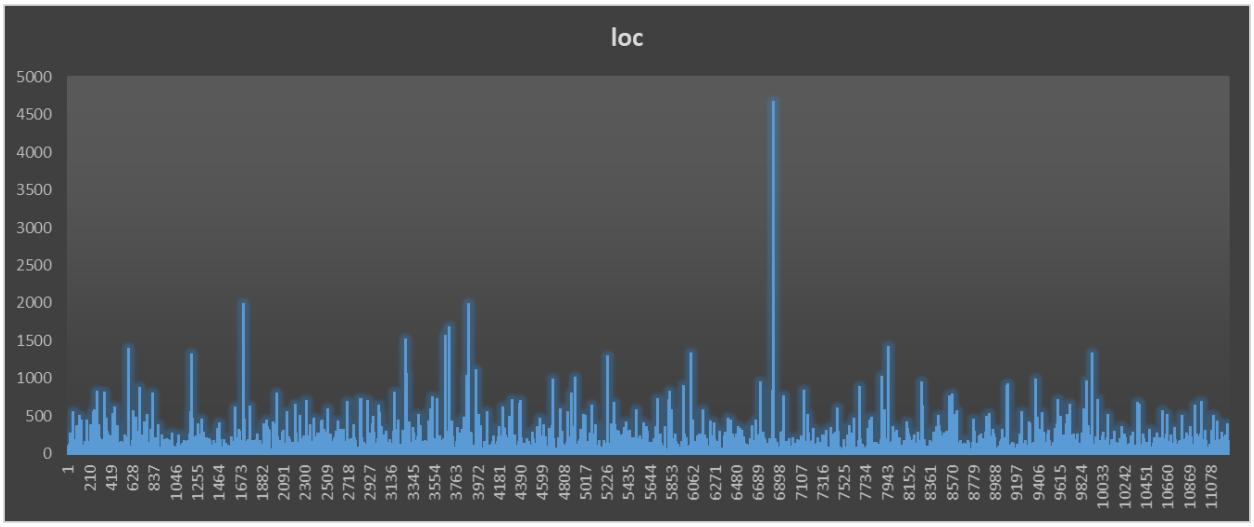


Project2: neo4j

	CBO	**RFC**	**LOC**
Max	147	275	4670
Mode	2	0	5
Median	4	3	19
Std Deviation	8.30141	19.00052	108.6678
Average	6.452292	10.5677	50.03962

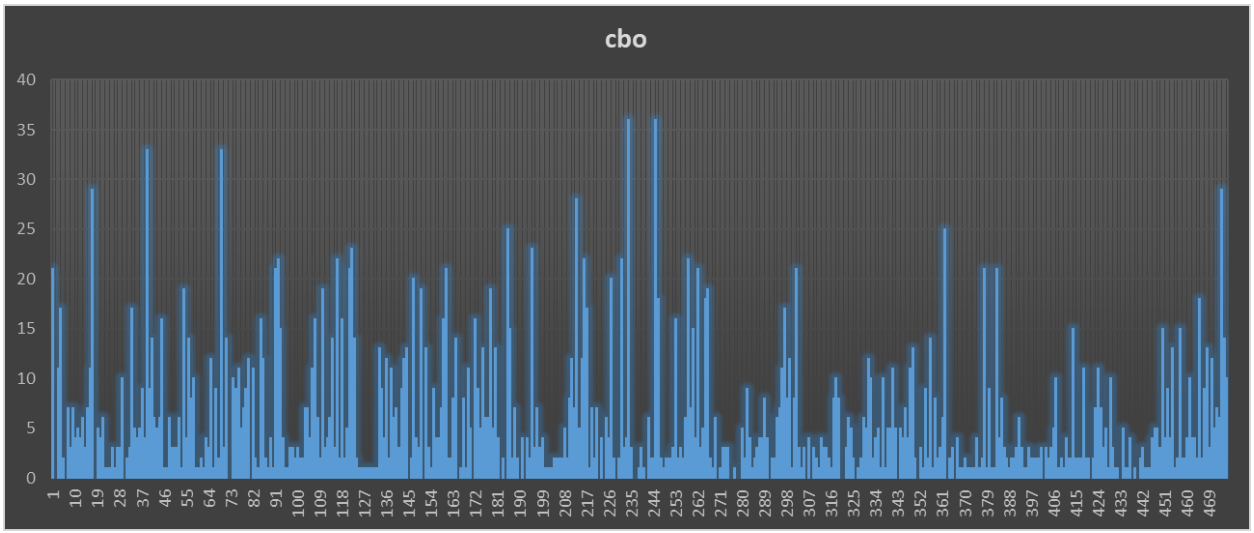

cbo
160
140
120
100
80
60
40
20
0
1
214
427
640
853
1066
1279
1492
1705
1918
2131
2344
2557
2770
2983
3196
3409
3622
3835
4048
4261
4474
4687
4900
5113
5326
5539
5752
5965
6178
6391
6604
6817
7030
7243
7456
7669
7882
8095
8308
8521
8734
8947
9160
9373
9586
9799
10012
10225
10438
10651
10864
11077

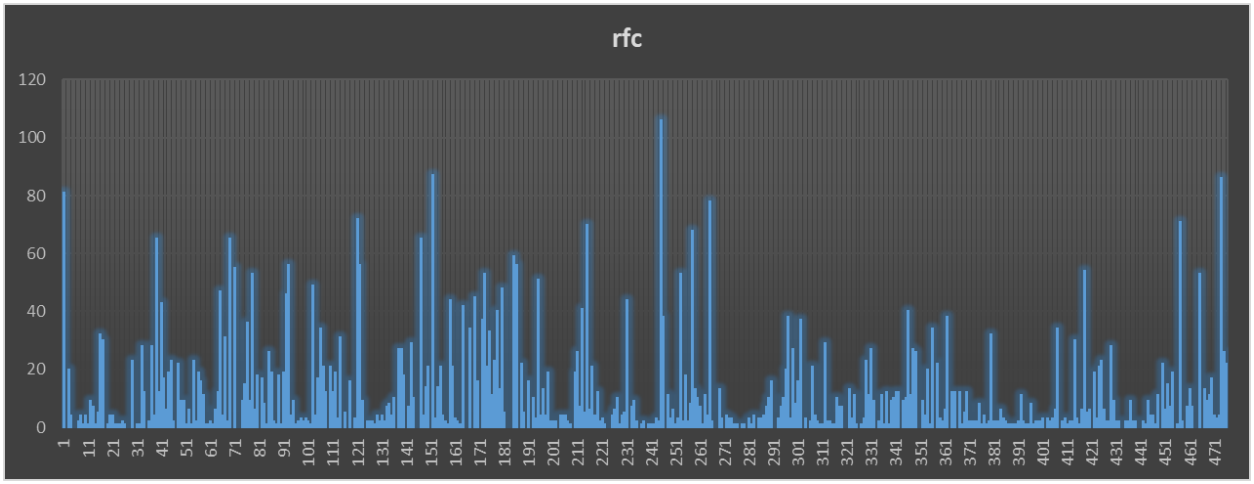

rfc
300
250
200
150
100
50
0
1
214
427
640
853
1066
1279
1492
1705
1918
2131
2344
2557
2770
2983
3196
3409
3622
3835
4048
4261
4474
4687
4900
5113
5326
5539
5752
5965
6178
6391
6604
6817
7030
7243
7456
7669
7882
8095
8308
8521
8734
8947
9160
9373
9586
9799
10012
10225
10438
10651
10864
11077

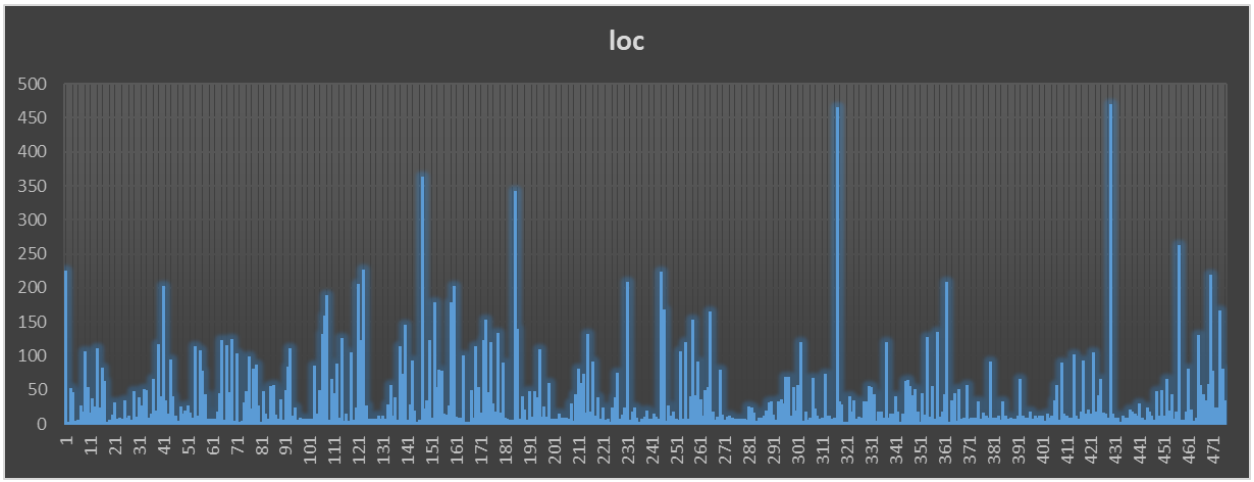


Project3 : Priam

	CBO	**RFC**	**LOC**
Max	36	106	468
Mode	2	0	6
Median	3	4	16
Std Deviation	6.424575	16.41774	57.42571
Average	5.852941	11.03151	39.78151

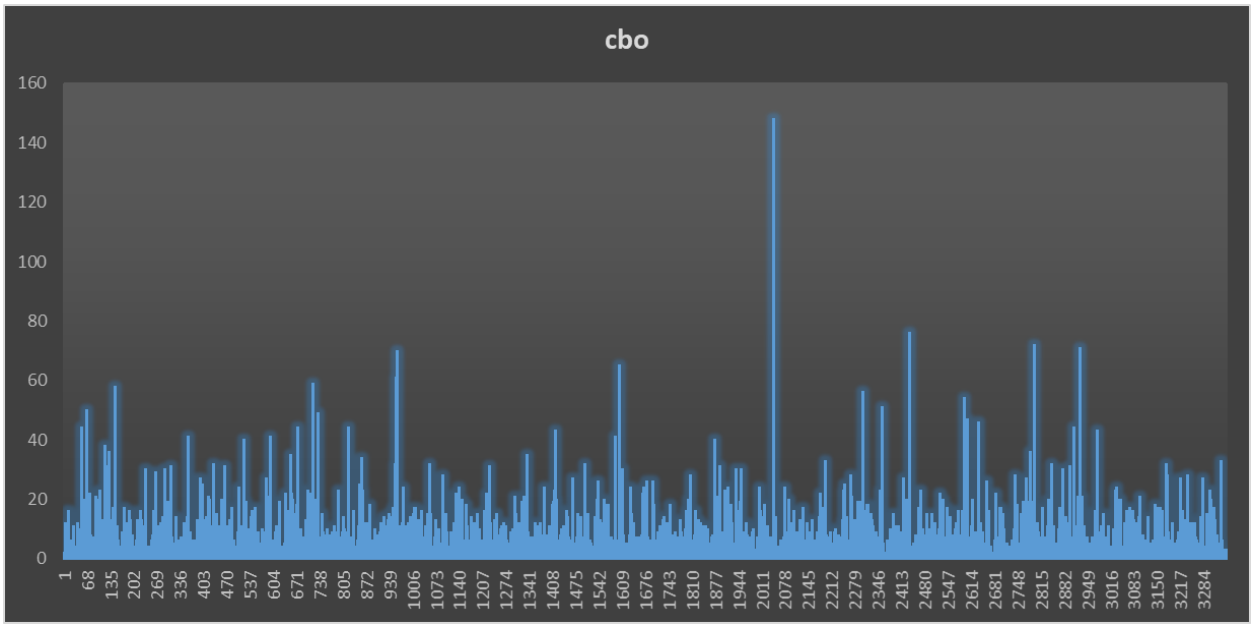


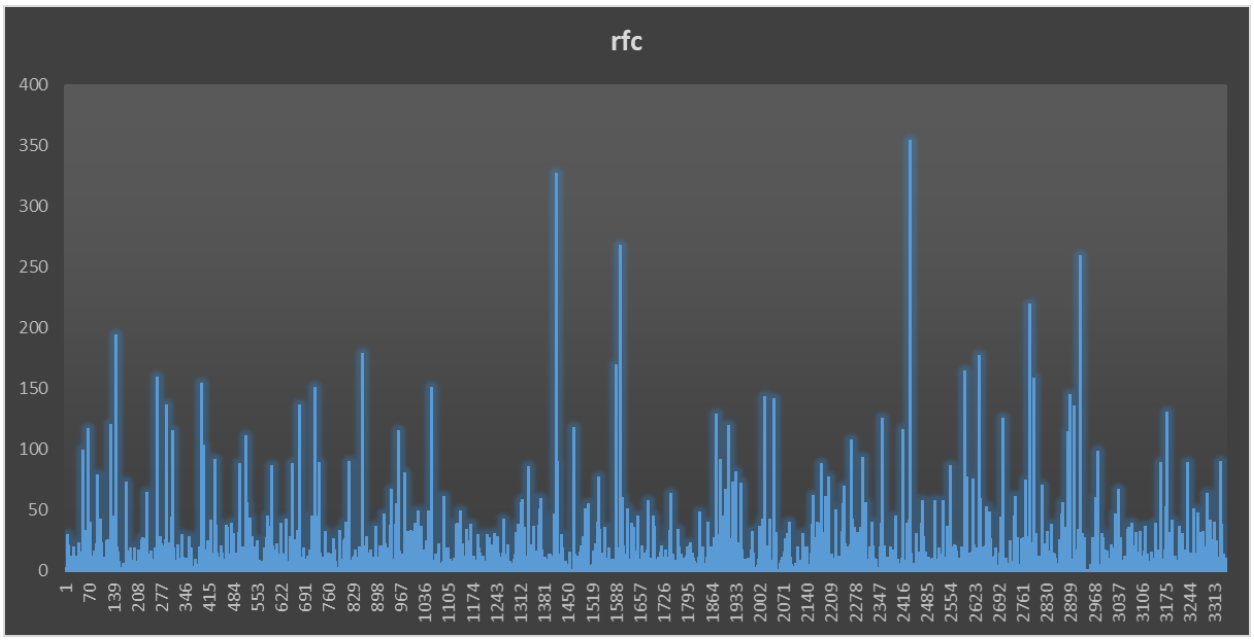


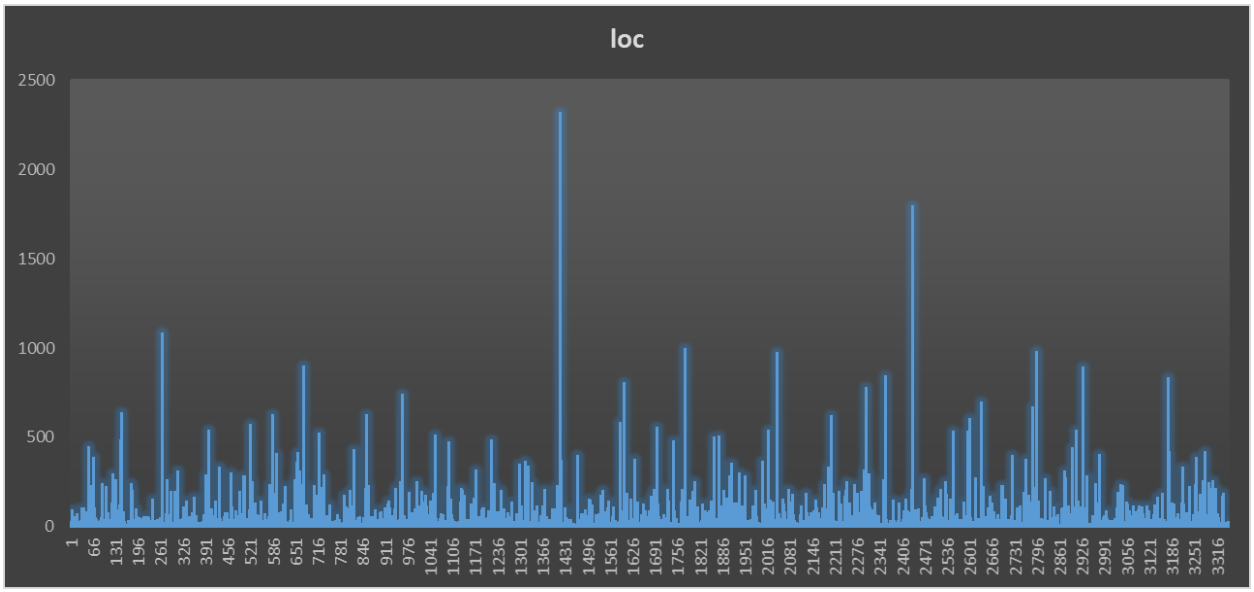


Project4: glowroot

	CBO	RFC	LOC
Max	148	354	2318
Mode	1	0	5
Median	3	3	12
Std Deviation	7.398608	21.13084	100.397
Average	5.341398	8.866487	40.30615



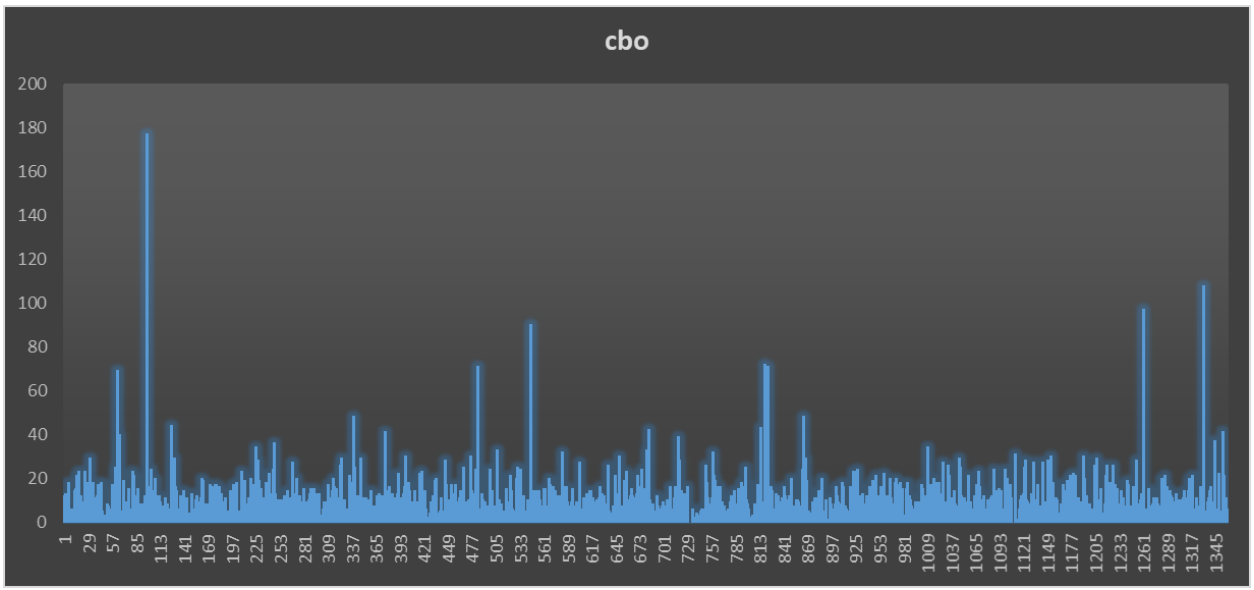


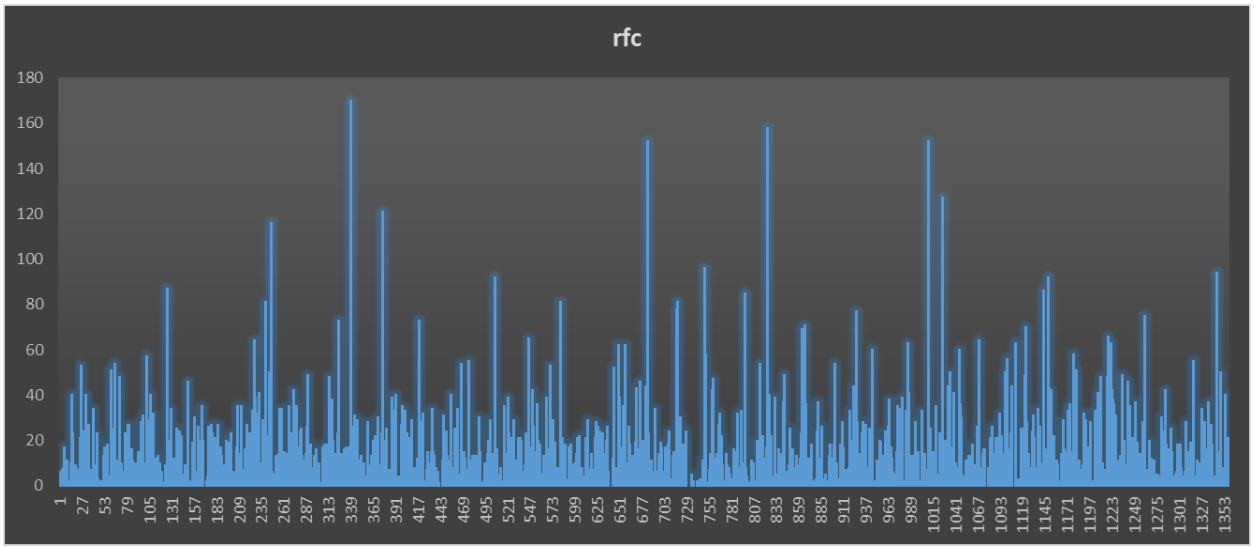


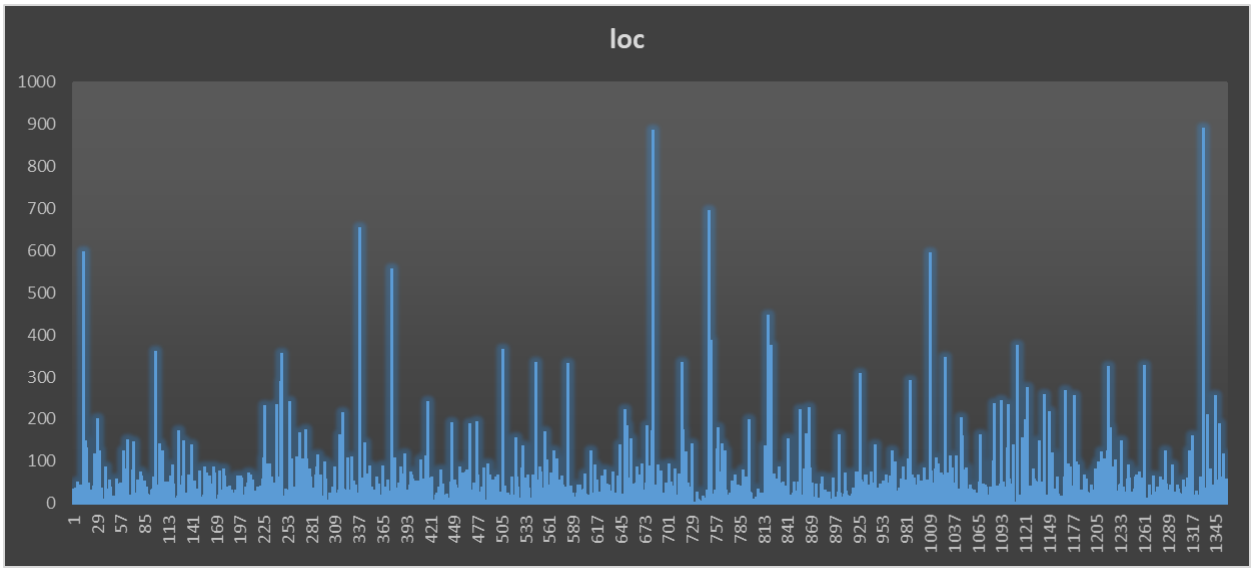
Project 5: Botania

	CBO	**RFC**	**LOC**
Max	177	170	891
Mode	6	0	8

Median	7	7	23
Std Deviation	10.06131	17.99074	72.30099
Average	9.412371	12.8299	44.40059







Section 5: Conclusion

In this research project, as a sole member of the group, I conducted an in-depth analysis to explore the relationship between size and maintainability in Java projects. By employing the CKJM tool to gather C&K metrics measurements and utilizing the Goal-Question-Metric (GQM) framework, I aimed to provide valuable insights for developers and organizations seeking to enhance their software development processes. Based on the analysis of the C&K metrics measurements, several key conclusions can be drawn regarding the correlation between size and maintainability in Java projects. Firstly, it was observed that larger classes, as measured by Lines of Code (LoC), tend to exhibit greater coupling (CBO) and higher response for a given class (RFC). This finding suggests a positive correlation between class size and interconnectivity with other classes, as well as the complexity of managing methods within a class. Increased coupling and RFC levels can pose challenges to maintainability, as modifications in one class may have a ripple effect on multiple others, and managing a large number of methods can lead to increased complexity.

However, the relationship between class size and Tight Class Cohesion (TCC) is not consistently observed. In some cases, a positive correlation between size and cohesion

is observed, while in others, a negative correlation is found. This suggests that the correlation between size and cohesion is intricate and influenced by various factors, such as development methodologies, code structuring, and the specific application domain. Furthermore, the impact of size on maintainability exhibits variability across different Java projects. Some projects demonstrate proficiency in managing maintainability challenges associated with size, while others face difficulties in maintaining code quality as project size expands.

This observation emphasizes the influence of project-specific factors, including development methodologies, developer expertise, and the complexity of the application domain, on the correlation between size and maintainability in Java projects. Based on these findings, it is advisable for developers and organizations to exercise caution when considering the size of their Java projects and anticipate the potential effects on maintainability. Larger projects, particularly those with high coupling and high RFC, may pose greater challenges in terms of maintainability. To address these challenges, developers should employ strategies such as modular design, efficient code organization, and adherence to design principles that promote low coupling and high cohesion.

However, further investigation is warranted to gain a deeper understanding of the underlying factors contributing to the observed correlations between size and maintainability in Java projects. Future research could explore the impact of development practices, code organization techniques, and the complexity of the application domain on maintainability. Additionally, it would be valuable to identify optimal strategies for mitigating size-related obstacles in Java projects.

In conclusion, the analysis of the selected Java projects in this research project reveals a correlation between size and maintainability. Larger projects tend to exhibit increased coupling and higher RFC, indicating potential challenges in managing maintainability. However, the relationship between size and cohesion is not straightforward and depends on various factors. Developers and organizations should consider these findings when undertaking Java projects, as understanding the correlation between size

and maintainability can guide the adoption of effective development methodologies and contribute to the long-term success of software projects.

References

Chidamber, S. R., & Kemerer, C. F. (1994). A metrics suite for object oriented design. IEEE Transactions on Software Engineering, 20(6), 476–493.https://doi.org/10.1109/32.295895

Basili, V., Caldiera, G., & Rombach, H. (n.d.). THE GOAL QUESTION METRIC APPROACH. https://www.cs.umd.edu/users/mvz/handouts/gqm.pdf

Software Metrics: A Rigorous and Practical Approach, Third Edition. (n.d.). Routledge & CRCPress.RetrievedApril10,2023,from https://www.routledge.com/Software-Metrics-A-Rigorous-and-Practical-Approach-Third-Edition/Fenton-Bieman/p/book/9780367659028

Martin, R. (n.d.). Agile Software Development, Principles, Patterns, and Practices. https://dl.ebooksworld.ir/motoman/Pearson.Agile.Software.Development.Principles.Patterns.and.Practices.www.EBooksWorld.ir.pdf

Design Patterns: Elements of Reusable Object-Oriented Software. (2019). O'Reilly | Safari. https://learning.oreilly.com/library/view/design-patterns-elements/0201633612/

Object-Oriented Metrics in Practice. (2006). Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-39538-5

Eclipse JDT Language Server. (2023, April 9). GitHub. https://github.com/eclipse/eclipse.jdt.ls

IntelliJ Platform SDK Documentation. (2023, April 9). GitHub. https://github.com/JetBrains/intellij-sdk-docs

Intra. (2023, April 7). GitHub. https://github.com/Jigsaw-Code/Intra

Apache/opennlp. (2020, September 9). GitHub. https://github.com/apache/opennlp

Netflix/Priam. (2023, March 15). GitHub. https://github.com/Netflix/Priam