Assignment 1 FINAL REPORT

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

Group-7
Pranavi Mittapalli

Akshith Paspula

Section 1 : Introduction

In this research paper we are going to show the relationship between the size of the java project and the maintainability that is related to them. The main goal of this project is to understand how the number of lines of code in a Java project can affect the maintainability. The research mainly shows the framework of Goal-Question-Metric whose work is to analyze and ensure. Java is an important programming language for the study purpose and its related application that is associated with it which has a wide range of use.The results that are obtained through Java programming can be applied to a wide range of projects and development teams include online applications, mobile apps and corporate systems. The Goal-Question-Metric methodology basically defines the research goals, formulate relevant questions and identify the appropriate metrics for analysis. The primary aim is to examine the relationship between Java project size and maintainability with the software development that can maintain the code.

To assess maintainability there are many metrics that are selected for the investigation. The metrics include the Lines of Code metrics , Coupling Between Objects , Response for a Class and Tight Class Cohesion . The metrics of Coupling Between Objects mainly measures the degree of interdependence between classes. When the maintainability is higher with lower coupling then it indicates that the complexity that is associated with the metrics is on a lower level. Response for a Class is a metric that counts the number of methods that can be executed in response to a message received by a class object. When the RFC value is low then it can have high maintainability. The smaller number of methods to be executed implies a more focused and cohesive class design. Tight Class

Cohesion is merices that can evaluate the level of interconnection between methods within a class. When the cohesion is higher then that means it indicates that the maintainability is stronger. The size of Java projects measured in LoC, the study aims to establish the correlation between project size and maintainability. The findings are expected to reveal trends and patterns that can guide the development of best practices for overseeing and sustaining large-scale Java projects.

Section 2 : Data set Description

Project Name	**Repository URL**	**Description**	**Size (LoC)**	**Developers**	**Age (Years)**
apollo	https://github.com/apolloconfig/apollo	Apollo is a reliable configuration management system suitable for microservice configuration management scenarios.	27561	143	8
ghidra	https://github.com/NationalSecurityAgency/ghidra	Ghidra is a software reverse engineering (SRE) framework	1697612	238	4

java-design-patterns	https://github.com/iluwatar/java-design-patterns	Design patterns implemented in Java	32211	347	2
mall	https://github.com/macrozheng/mall	The project is an e-commerce system consisting of a front-end mall system and a back-end management system, implemented using SpringBoot+MyBatis and deployed with Docker containers.	100691	1	1
spring-framework	https://github.com/spring-projects/spring-framework	Spring Framework	653907	734	8

Section 3 : Tool Description

In this research paper CKJM tool is basically utilized so that for the chosen project C&K mertices can measure the class . CKJM is a software tool that is used to operate with Java-based projects and perform the computation of Chidamber and Kemerer's metrics for object-oriented programming. CKJM is specifically designed to work with Java-based

projects and compute the C&K metrics. It includes Coupling Between Objects, Response for a Class and Tight Class Cohesion . The Metrics are directly relevant to the research inquiry and provide insights into the correlation between project size and maintainability. CKJM is a command-line utility that offers a user-friendly interface and easy installation and operation. It accepts Java bytecode files in the .class format and generates the desired metrics in a clear and concise manner. The user's familiarity with the tool allows for efficient acquisition of the required measurements without investing significant time in understanding complex instruments. CKJM is an open-source tool accompanied by a user guide . It also ensures transparency and facilitates users' understanding of its functionality and the metrics it provides. The transparent metrics acquisition process enhances the reliability and trustworthiness of the obtained measurements for analytical purposes. One of the advantages of CKJM is the flexibility that is associated with it as it allows the user to select the specific C&K metrics they wish to compute . It enables the user to choose the relevant metrics for the analysis. For understanding maintainability in Java projects the main thing that needs to focus on is the CBO, RFC and TCC.

The CKJM software metric tool is accessible with the subsequent Uniform Resource Locator : https://github.com/mauricioaniche/ck

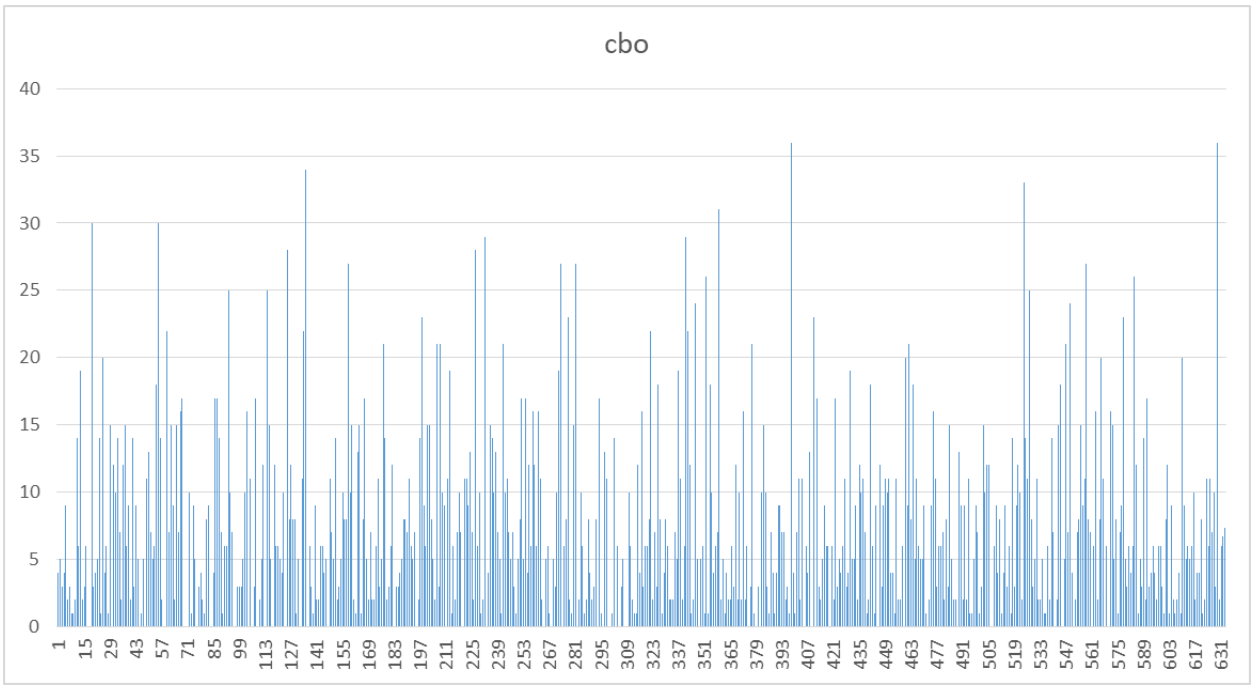
CKJM is a well-established and widely referenced tool that has been utilized in numerous academic investigations. To utilize CKJM for the analysis the first step involved retrieving the source code of each selected project from the GitHub platform. The projects were then compiled to generate the corresponding Java bytecode files (.class). Second step, the CKJM tool was used to obtain the C&K metrics measurements for each class. These metrics were employed to examine the correlation between project size and

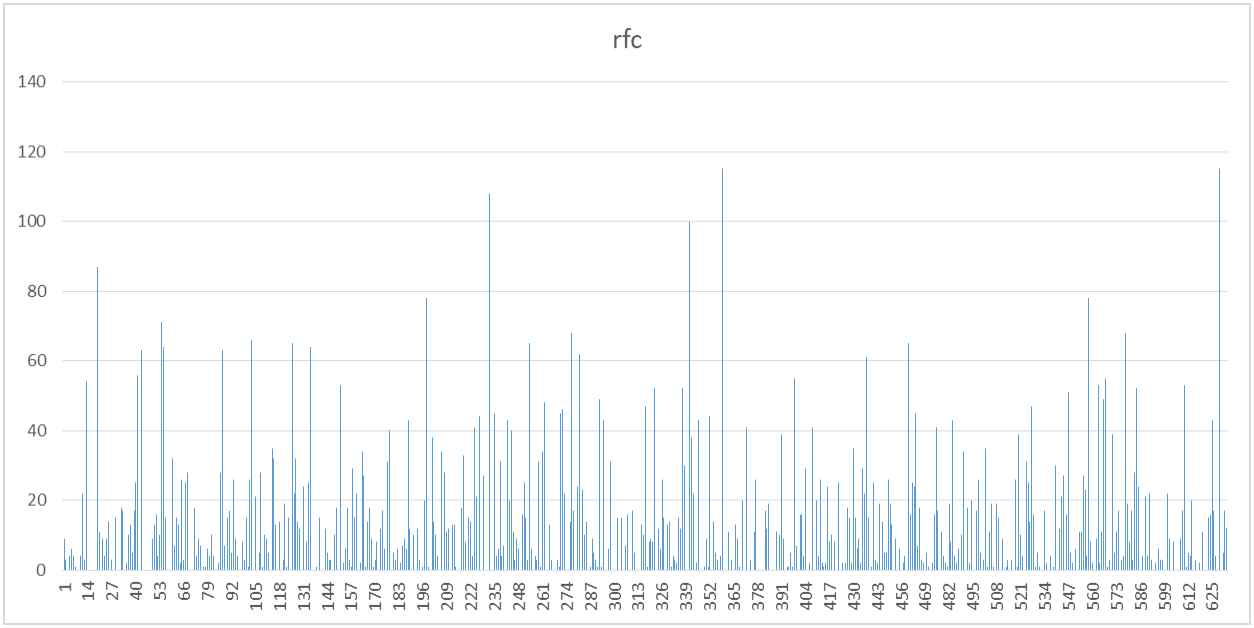
maintainability in the designated Java projects. Java being a user-friendly interface, open-source nature, transparency, flexibility in metric selection and established reputation within the research community make it a reliable instrument for this research paper.

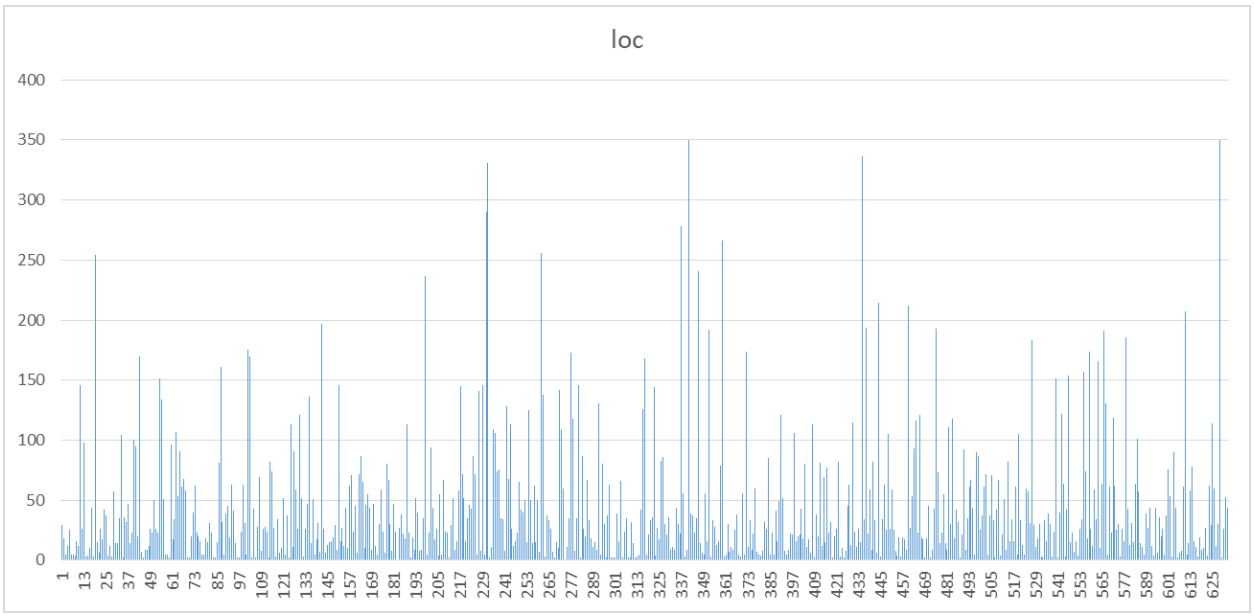
Section 4 : Project Results

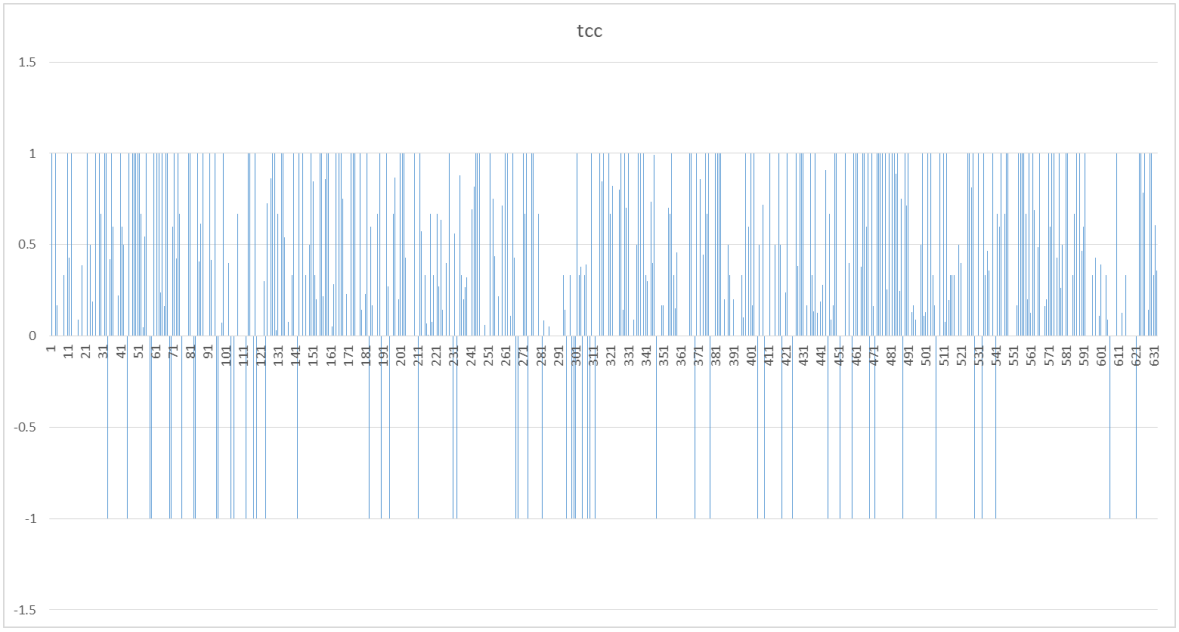
Project 1: apollo

	CBO	RFC	TCC	LOC
Max	36	115	1	350
Mode	2	0	1	2
Median	6	5	0.333333	26
Std Deviation	6.667597699	17.01130632	0.607208	52.40242657
Average	7.310509554	12.16878981	0.356834	43.88694268


cbo
40
35
30
25
20
15
10
5
0
1 15 29 43 57 71 85 99 113 127 141 155 169 183 197 211 225 239 253 267 281 295 309 323 337 351 365 379 393 407 421 435 449 463 477 491 505 519 533 547 561 575 589 603 617 631


rfc
140
120
100
80
60
40
20
0
1 14 27 40 53 66 79 92 105 118 131 144 157 170 183 196 209 222 235 248 261 274 287 300 313 326 339 352 365 378 391 404 417 430 443 456 469 482 495 508 521 534 547 560 573 586 599 612 625

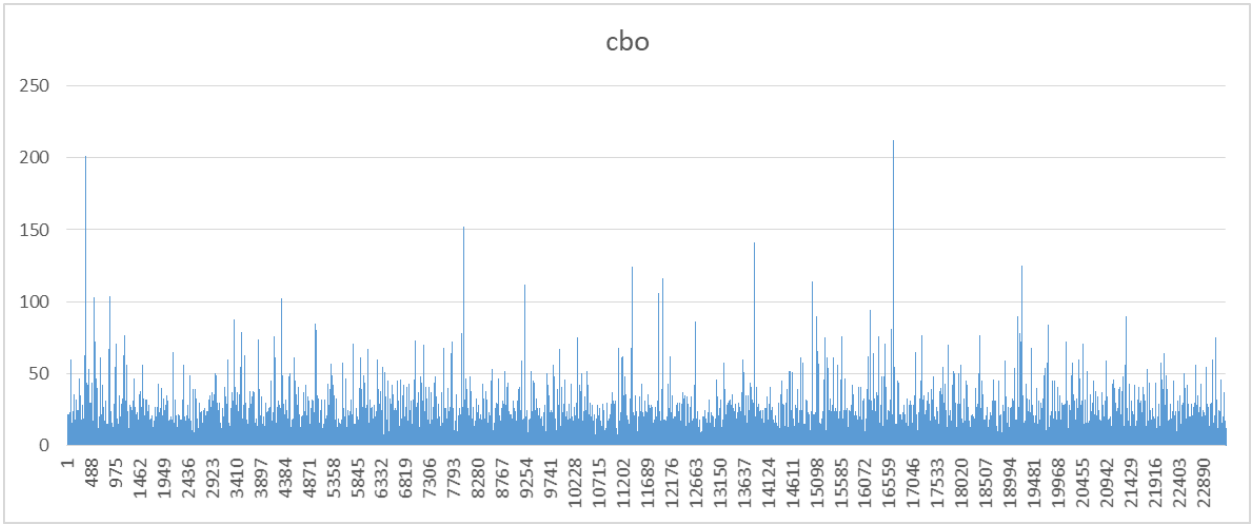


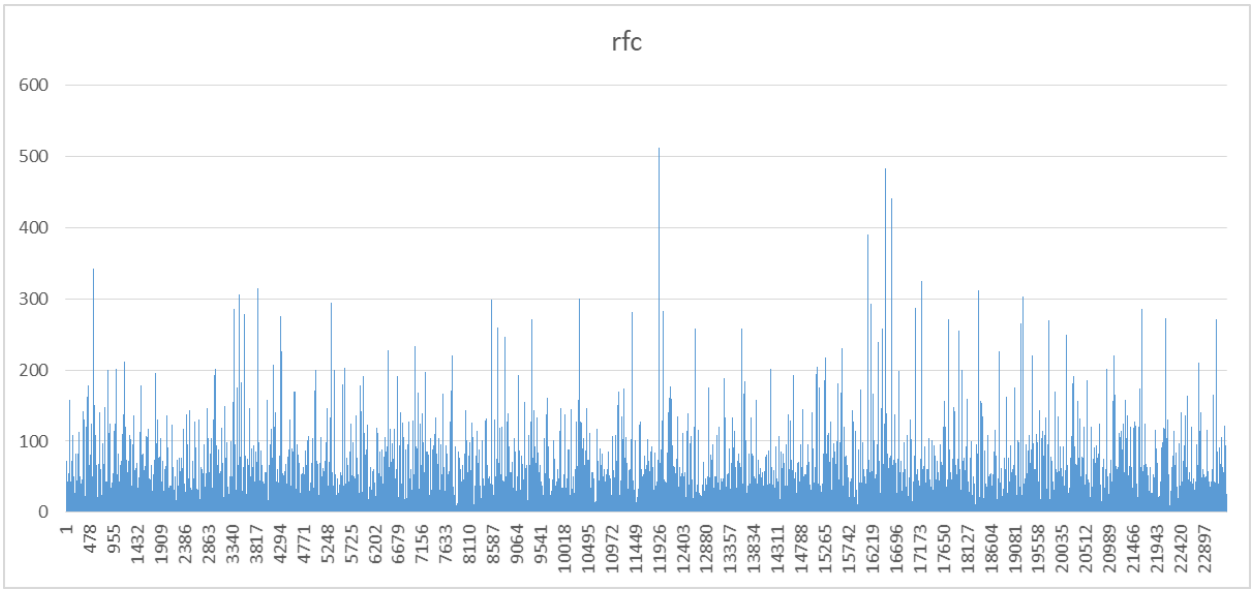


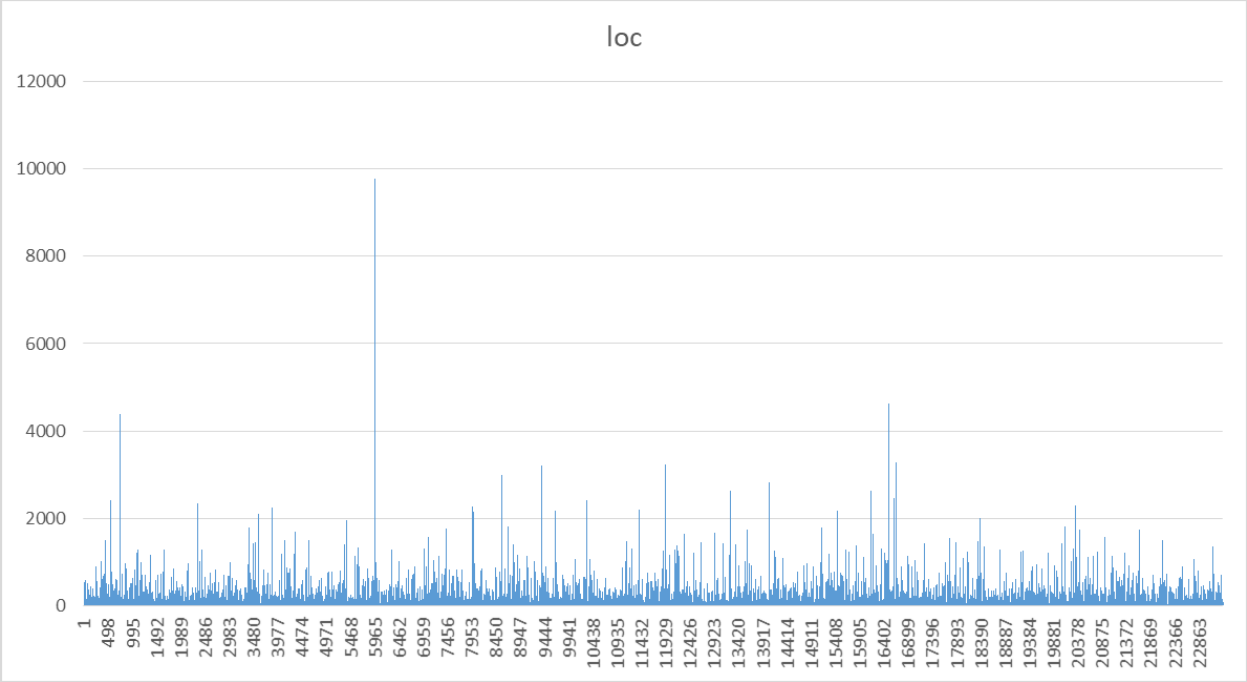
Project 2: ghidra

	CBO	**RFC**	**TCC**	**LOC**
Max	212	512	1	9774
Mode	2	0	0	5

Median	4	5	0.00952381	26
Std Deviation	8.806323764	25.26468581	0.441375846	174.8147493
Average	7.007323027	13.46229284	0.196278144	72.6997559

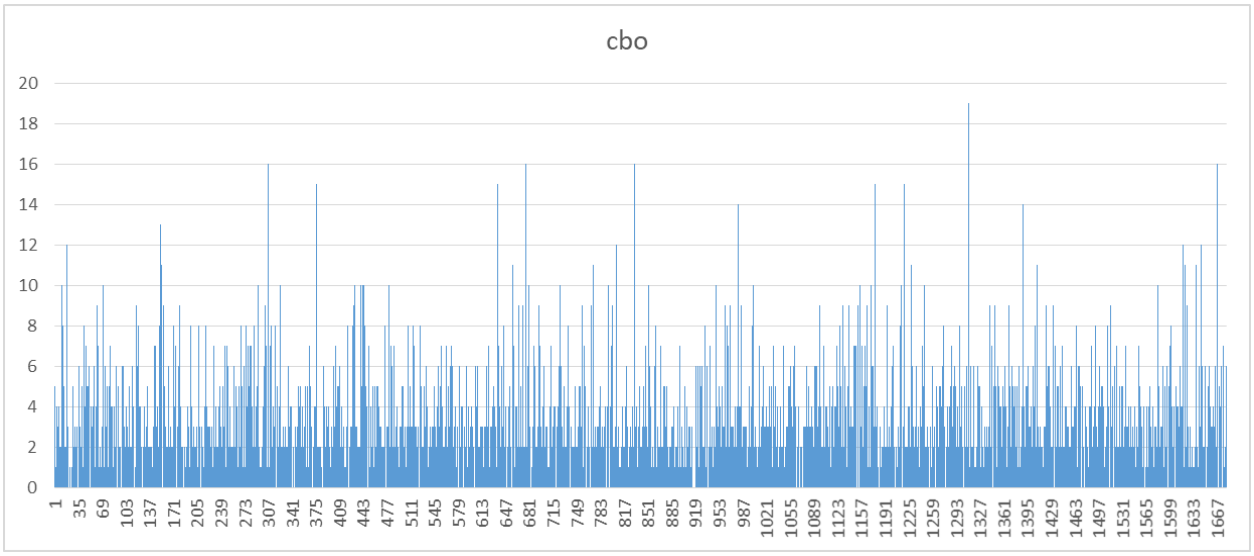


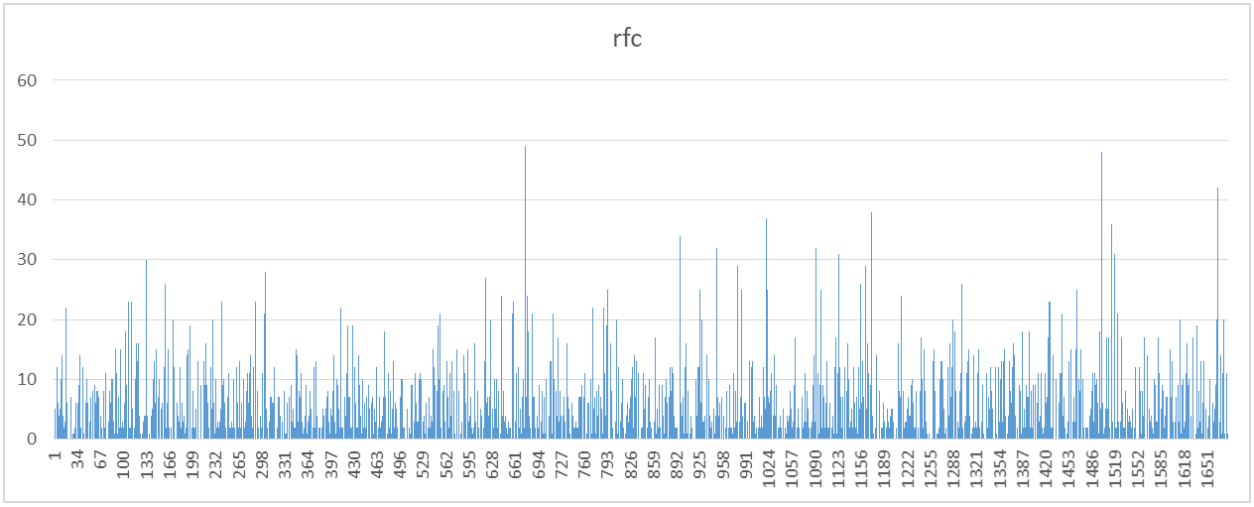


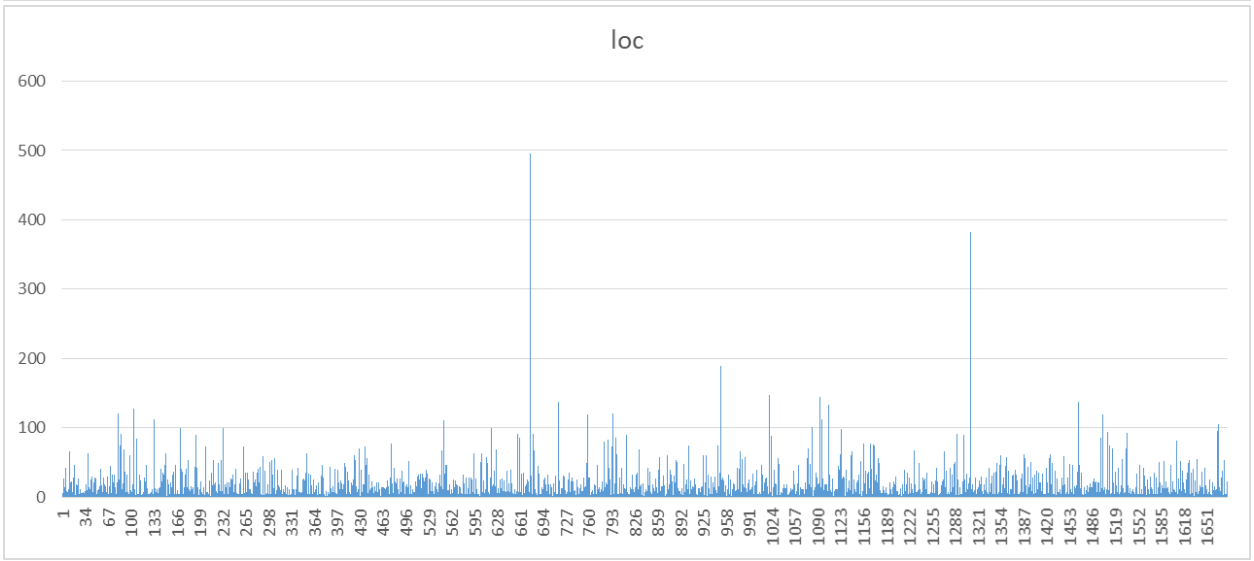


Project 3 : java-design-patterns

	CBO	RFC	TCC	LOC
Max	19	49	1	496
Mode	2	0	0	5
Median	3	3	0.06666667	12
Std Deviation	2.524117	5.971691	0.61382639	25.11259451
Average	3.23631	4.795833	0.266046893	19.17321429

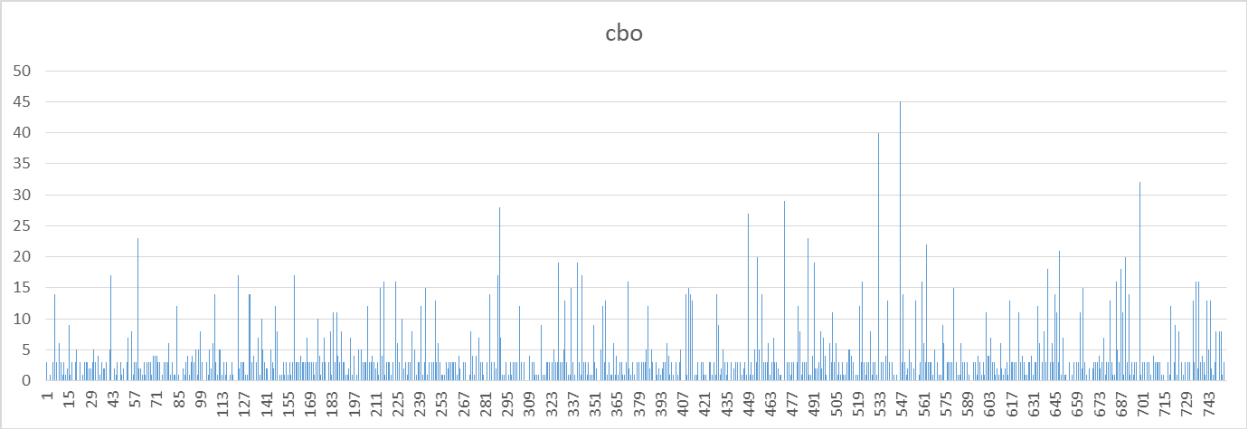


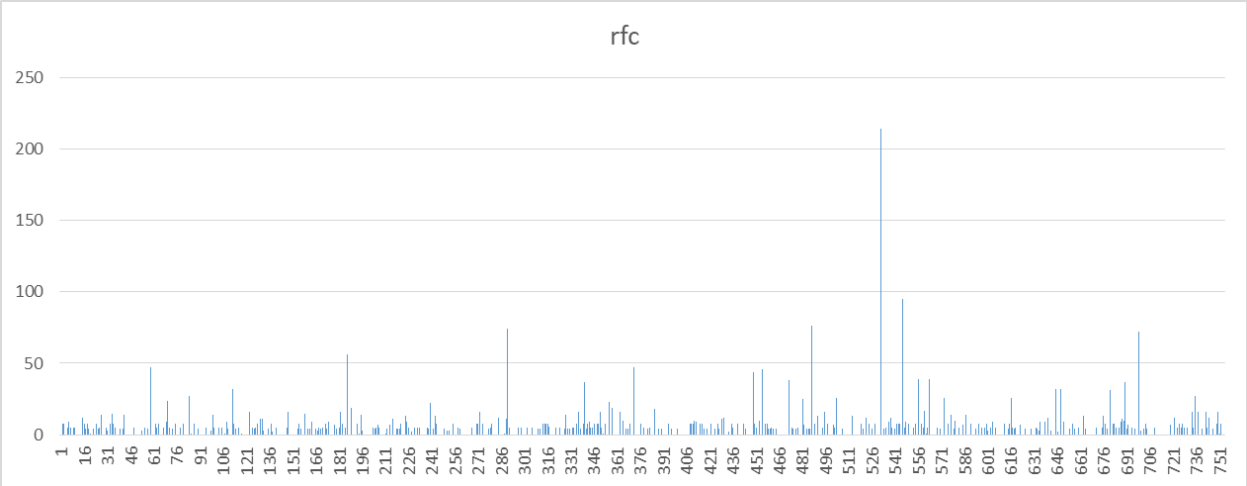


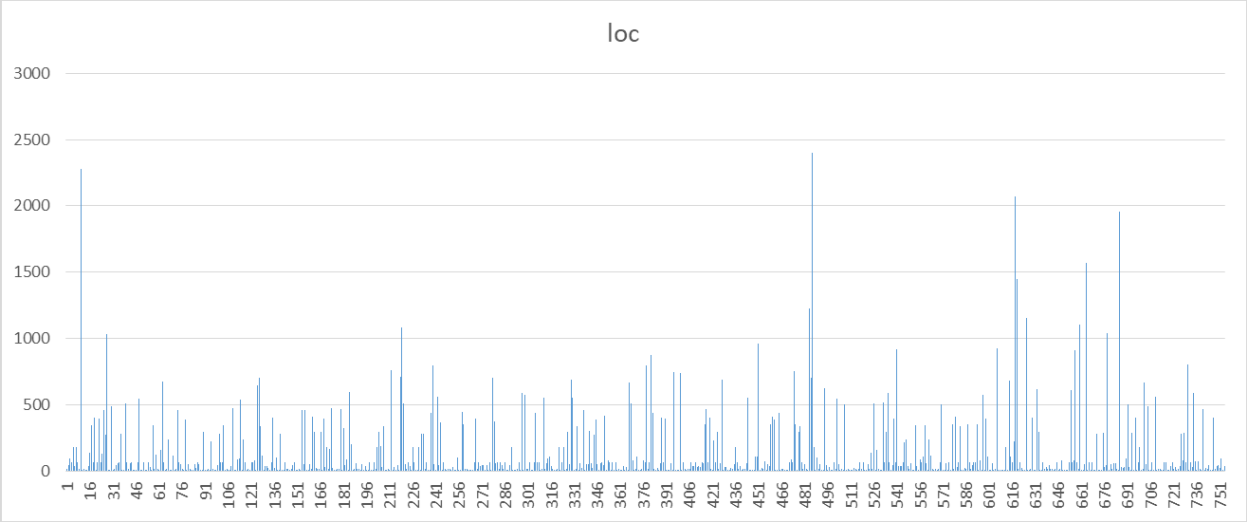


Project 4: mall

	CBO	RFC	TCC	LOC
Max	45	214	1	2400
Mode	3	0	0	5
Median	3	0	0.111111	32
Std Deviation	4.946221	11.64802	0.486797	261.9813
Average	3.912467	4.834218	0.146681	133.5424

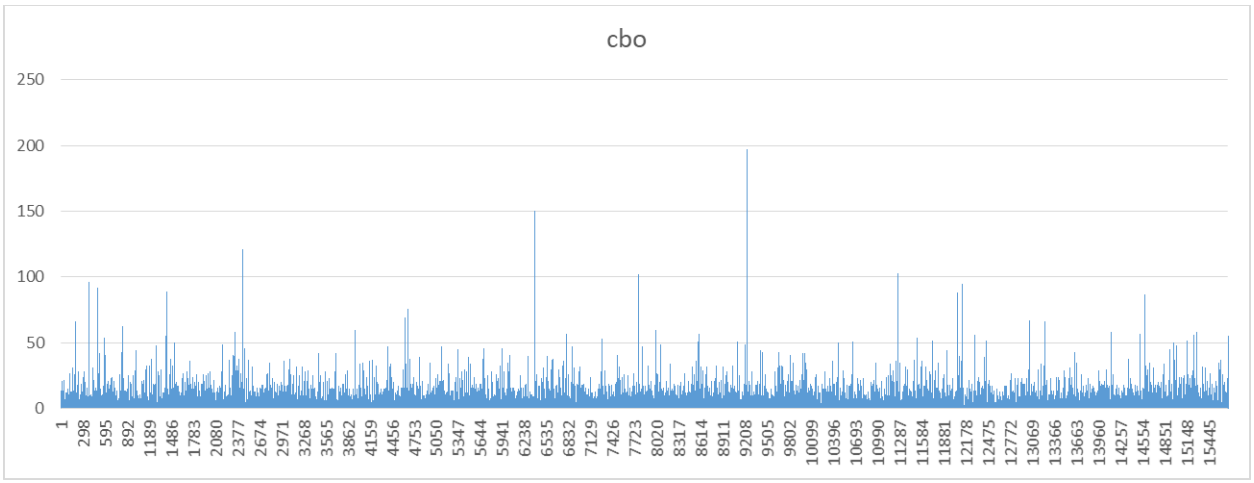


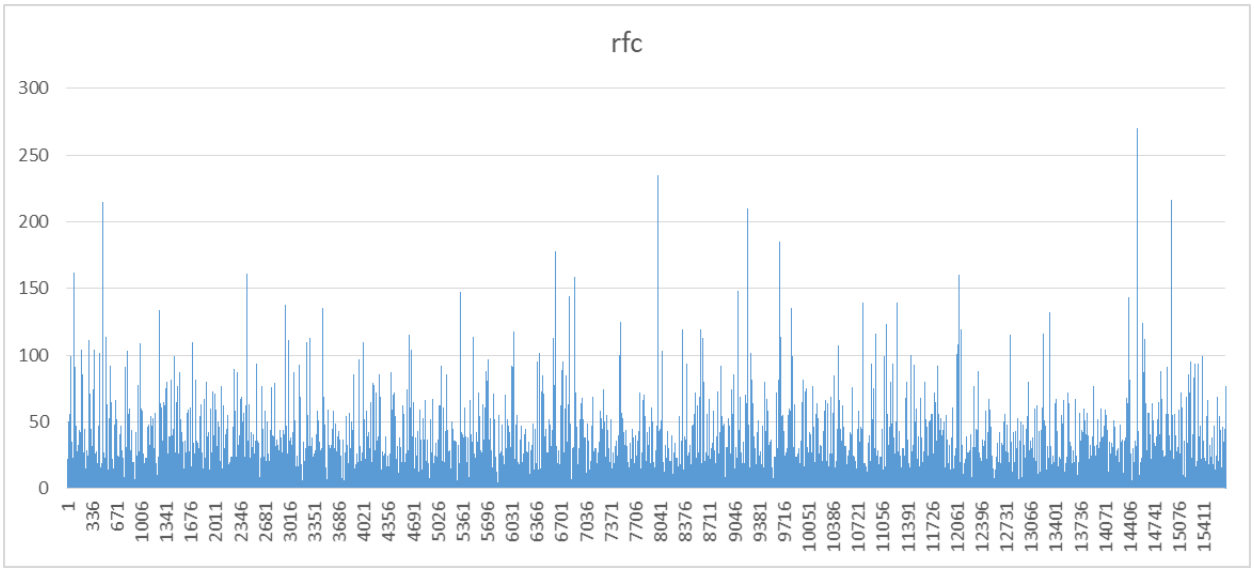


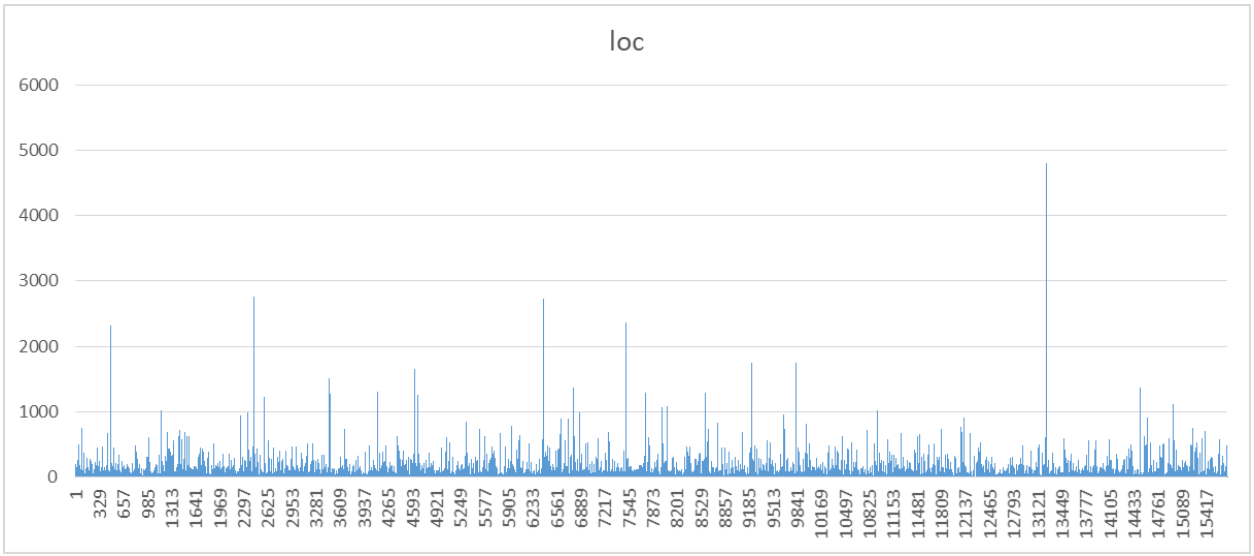


Project 5: spring-framework

	CBO	RFC	TCC	LOC
Max	197	270	1	4800
Mode	1	0	0	2
Median	3	2	0	12
Std Deviation	6.882079	15.65635	0.619358	104.34
Average	5.24215	8.257181	0.073941	41.64747


cbo
250
200
150
100
50
0
1
298
595
892
1189
1486
1783
2080
2377
2674
2971
3268
3565
3862
4159
4456
4753
5050
5347
5644
5941
6238
6535
6832
7129
7426
7723
8020
8317
8614
8911
9208
9505
9802
10099
10396
10693
10990
11287
11584
11881
12178
12475
12772
13069
13366
13663
13960
14257
14554
14851
15148
15445


rfc
300
250
200
150
100
50
0
1
336
671
1006
1341
1676
2011
2346
2681
3016
3351
3686
4021
4356
4691
5026
5361
5696
6031
6366
6701
7036
7371
7706
8041
8376
8711
9046
9381
9716
10051
10386
10721
11056
11391
11726
12061
12396
12731
13066
13401
13736
14071
14406
14741
15076
15411


loc
6000
5000
4000
3000
2000
1000
0
1
329
657
985
1313
1641
1969
2297
2625
2953
3281
3609
3937
4265
4593
4921
5249
5577
5905
6233
6561
6889
7217
7545
7873
8201
8529
8857
9185
9513
9841
10169
10497
10825
11153
11481
11809
12137
12465
12793
13121
13449
13777
14105
14433
14761
15089
15417

Section 5 : Conclusion

The research paper provides valuable information for developers and organizations aiming to improve their software development processes. An analysis of C&K metrics measurements derived from a set of Java projects was conducted to examine the correlation between project size and maintainability. The analysis mainly reveals larger classes that are measured by Lines of Code (LoC) that tend to exhibit greater coupling and increased response for a given class . This also creates a positive relation between the class size and the interconnectivity with the other class as a result the number of methods that can be executed within a class . When the coupling is higher then RFC can create a negative impact in the maintainability. When a single class requires modification then multiple classes are required to change and a large number of methods also can introduce the complexity . The relationship between Tight Class Cohesion and class size is not always observed. The positive relation between class size and cohesion exists while in other cases a negative relation is also observed. This also shows that the relation between the size and cohesion is complex. It is also dependent on the variable like the development methodologies, code structure and field of application.

The effect of size on maintainability also varies across different projects. Some projects demonstrate proficiency in handling maintainability challenges associated with size, while others encounter difficulties in maintaining code quality as the project size expands. This indicates project-specific factors like development methodologies, the proficiency of developers and the complexity of the application domain. It can significantly impact the correlation between size and maintainability in Java projects. Large projects like those with high coupling and high RFC really can pose greater difficulties in maintenance. To

address the challenges, developers should employ strategies such as modular design, efficient code organization and adherence to design principles that promote low coupling and high cohesion. This research paper has the potential to explore the impact of development practices, code organization and application domain complexity on maintainability. To gain a deeper understanding of the factors contributing to the observed correlations between size and maintainability in Java projects.

Reference :

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

Achtaich, A., Rowdies, O., Souissi, N., Salinesi, C., & Mazo, R. (2019, September). Evaluation of the state-constraint transition modeling language: A goal question metric approach. In *Proceedings of the 23rd International Systems and Software Product Line Conference-Volume B* (pp. 106-113).

Tuah, N. M., & Wills, G. B. (2020). Measuring the application of anthropomorphic gamification for transitional care; a goal-question-metric approach. In *Computational Science and Technology: 6th ICCST 2019, Kota Kinabalu, Malaysia, 29-30 August 2019* (pp. 553-564). Springer Singapore.

Utomo, R. G., Wills, G., & Walters, R. (2021). Developing an instrument to measure information assurance implementation for eGovernment using goal question metric

approach. *Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control*.