**Group Project – 1**

Jaya Sainatha Reddy Yerragudi

Lakshmi Prasanna Potla

Kaushik Gangireddy

Lewis University

Object Oriented Development

November 9, 2023

**Objective**

Perform an empirical study with the objective of finding the effect of class size on software maintainability

**Question**

"What is the impact of class size on software maintainability?"

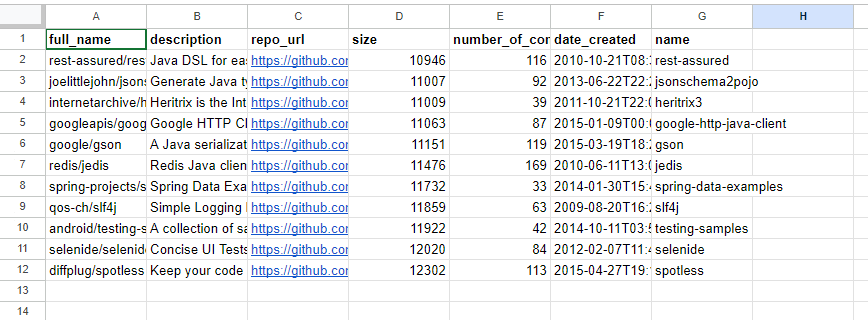
**Metrics**

Dependent Variables - CBO (Coupling Between Objects), LCOM\* (Lack of Cohesion of Methods)

Independent variable – LOC (Lines of code)

**Criteria**

We would like to analyze 11 Java programs that are at least 10K in size, 5 years old and have at least 5 developers.



***Justification for Criteria***

The choice of Java programs for analysis makes sense because we're looking at larger projects (over 10K in size) that have been around for at least 5 years and involve at least 5 developers. The large codebase allows us to dive into complex architectural and coding aspects, while the age of the projects suggests they've been through various changes and updates, giving us insights into long-term stability. Having a team of at least 5 developers brings in the collaborative element, helping us understand how different team members contribute, communicate, and follow coding standards.

**Tools Used**

We have used the CK Metrics tool to get the metrics and used JASP to perform correlation for analysis.

**Results**

The metrics have already been uploaded to the git repository for reference.

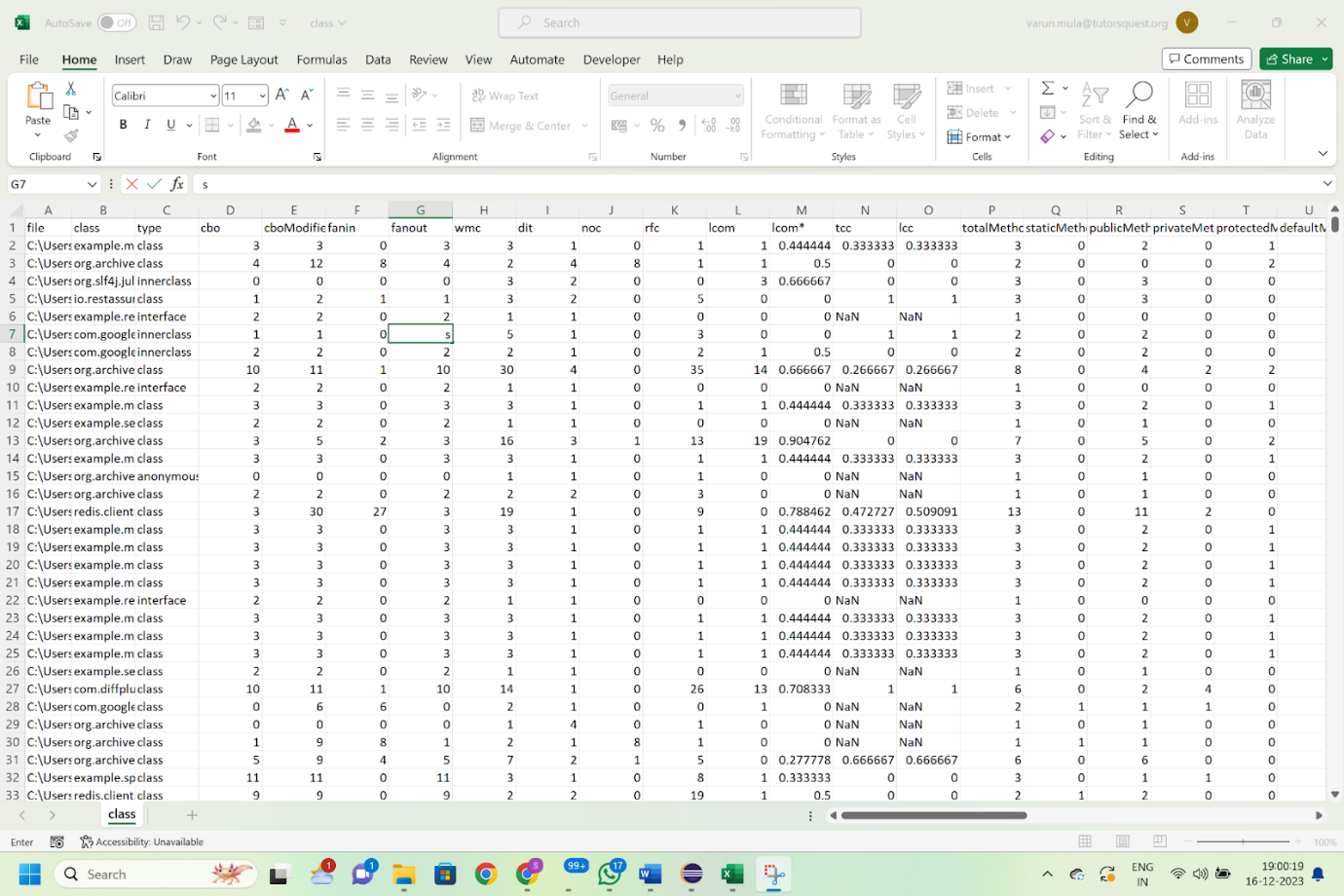


Figure: Metrics obtained using CK tool

A screenshot of a computer

Description automatically generated

Figure - Scatter plot of cbo vs loc

A screenshot of a computer

Description automatically generated

Figure - scatter plot of lcom vs loc

**Analysis**

The correlation coefficient (Pearson's r) between "cbo" and "loc" is 0.681. This positive correlation indicates a moderate to strong positive relationship between the number of lines of code and the coupling between objects. In other words, as the lines of code increase, there tends to be an increase in the coupling between objects. The p-value being less than 0.001 suggests that the observed correlation is statistically significant. In practical terms, this means that it's unlikely to have occurred by random chance. Therefore, you have evidence to reject the null hypothesis that there is no correlation between "cbo" and "loc."

The correlation coefficient (Pearson's r) between "lcom" (lack of cohesion in methods) and "loc" (lines of code) is 0.304. This positive correlation suggests a weak to moderate positive relationship between the lack of cohesion in methods and the number of lines of code. In simpler terms, as the lines of code increase, there tends to be a slight to moderate increase in the lack of cohesion in methods. The p-value being less than 0.001 indicates that this correlation is statistically significant. This means that the observed correlation is unlikely to have occurred by random chance, providing evidence to reject the null hypothesis that there is no correlation between "lcom" and "loc."

**Conclusion**

The identified correlations hint at a notable concern when dealing with larger codebases. As code size grows, the intricate relationships between objects and the potential lack of cohesion within methods pose inherent challenges to maintainability. This complexity can create a codebase that resists modifications, potentially impacting the agility and efficiency of the development process.

***Recommendations for Improvement***

**Enhancing Cohesion Within Methods:** Refactoring endeavors should strategically prioritize the enhancement of cohesion within methods (Tiwari & Rathore, 2018). By doing so, the overall structure and readability of the code can be improved. This involves organizing related functionalities within the same methods, fostering a more coherent and comprehensible codebase.

**Reducing Coupling Between Objects:** Strategic approaches to diminish coupling between objects become imperative. This involves implementing strategies to minimize the dependencies between different classes. By doing this, the ripple effects of changes can be mitigated, contributing to a more adaptable system over time.

In simpler terms, the bigger the codebase, the trickier it becomes to maintain. To tackle this, it's crucial to tidy up the internal structure of the code, making sure each piece does its job well and is easy to understand. Additionally, efforts should be made to untangle the connections between different parts of the code, allowing for smoother adjustments without causing a domino effect of changes.

**References**

<https://github.com/mauricioaniche/ck>

Goss-Sampson, M. (2019). Statistical analysis in JASP: A guide for students.

Tiwari, S., & Rathore, S. S. (2018, February). Coupling and cohesion metrics for object-oriented software: A systematic mapping study. In Proceedings of the 11th Innovations in Software Engineering Conference (pp. 1-11).