Group-Assignment1

First empirical study: Effect of class size on software maintainability

GQM Approach

The Goal-Question-Metric (GQM) approach is a systematic and structured method used for defining, measuring, and achieving goals in various domains, including software engineering, project management, and quality assurance. Based upon the assumption that a study to measure in a purposeful way it must first specify the goals, then it must trace those goals to the data that are intended to define those goals.

GQM defines a measurement model on three levels:

Conceptual level (Goal) A goal is defined for an object, for a variety of reasons, with respect to various models of quality, from various points of view and relative to a particular environment.

Operational level (Question) A set of questions is used to define models of the object of study and then focuses on that object to characterize the assessment or achievement of a specific goal.

Quantitative level (Metric) A set of metrics, based on the models, is associated with every question in order to answer it in a measurable way

Objectives: Used the below listed Projects to measure the complexity of the classes with the help of C&K Metrics tool and to identifying the ways improve code quality and software maintainability for the selected projects.

1. Free Universal Database Tool and SQL Client (DBeaver)

2. Generate diagrams from textual description

3. Graphs for everyone

4. OpenPDF is a free Java library for creating and editing PDF files with a LGPL and MPL open-source license.

5. Provide support to increase developer productivity in Java when using MongoDB.

GQM Approach for Dbeaver: Free Universal Database Tool and SQL Client (DBeaver)

Developing the Set of Goals: The primary objective is to assess and improve the quality and usability of the DBeaver open-source project.

Questions That Define the Goals in a Quantifiable Way:

Question 1: What is the current quality and usability level of the DBeaver project according to its users?

Question 2: What specific aspects of DBeaver need improvement to enhance the user experience?

Question 3: What factors are contributing to maintainability issues in specific classes of the DBeaver project

Question 4: Are there specific classes that require refactoring or optimization to improve maintainability?

Metrics - Specify the Measures That Need to Be Collected to Answer Those Questions:

Metric 1: Weighted Methods per Class (WMC): Measures the complexity of classes in terms of the number of methods.

Metric 2: Depth of Inheritance Tree (DIT): Indicates how deep in the inheritance hierarchy a class is.

Metric 3: Number of Children (NOC): Quantifies the number of child classes inheriting from a parent class.

Metric 4: Response for a Class (RFC): Measures the number of methods that can be executed in response to a message to the class.

Metric 5: CBO: Coupling Between Objects: count of the number of other classes to which a class is coupled.

Metric 6: LCOM: Lack of Cohesion of Methods: measures the dissimilarity of methods in a class by instance variable or attributes.

Develop Mechanisms for Data Collection:

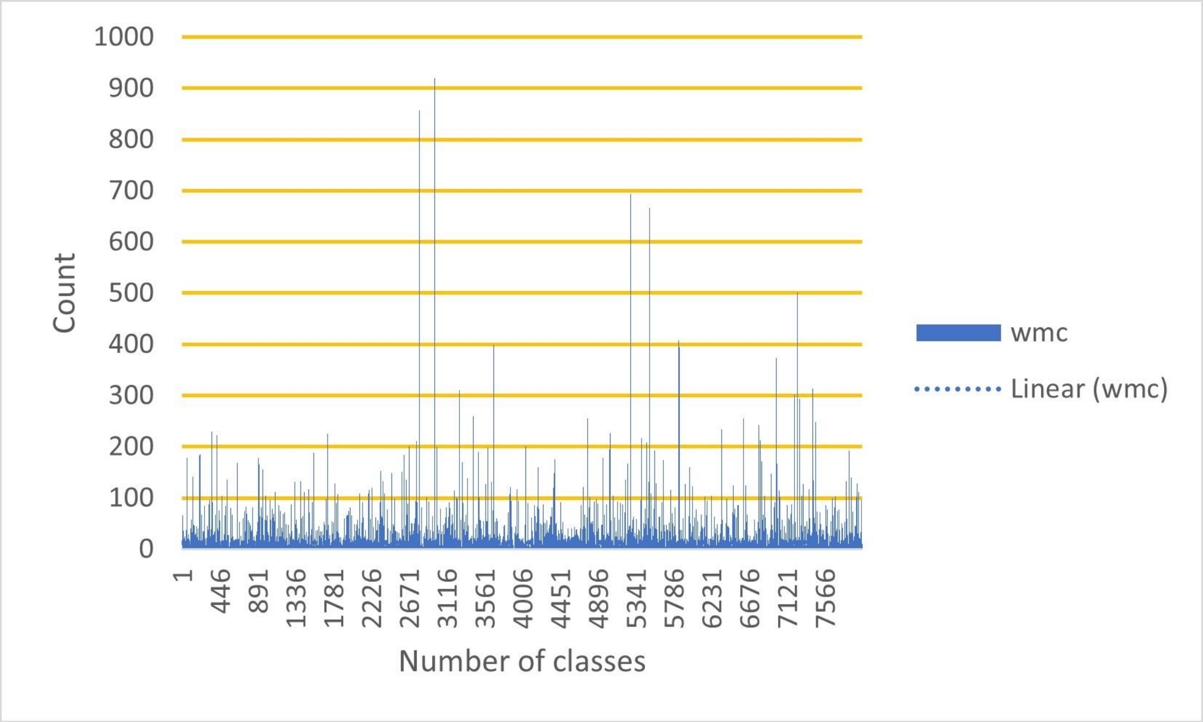
By using C&K Metrics tool, Collected the metrics information for the classes.

The Chidamber and Kemerer (C&K) metrics, also known as the CK metrics suite, is a set of software complexity metrics used to evaluate the structural complexity of object-oriented software systems. These metrics were introduced by Shyam R. Chidamber and Chris F. Kemerer in their paper titled "A Metrics Suite for Object-Oriented Design" in 1994.

CK calculates class-level and method-level code metrics in Java projects by means of static analysis (i.e. no need for compiled code).

WMC: Maximum number of methods for in a Class is 920

Minimum number of methods in a class is 0



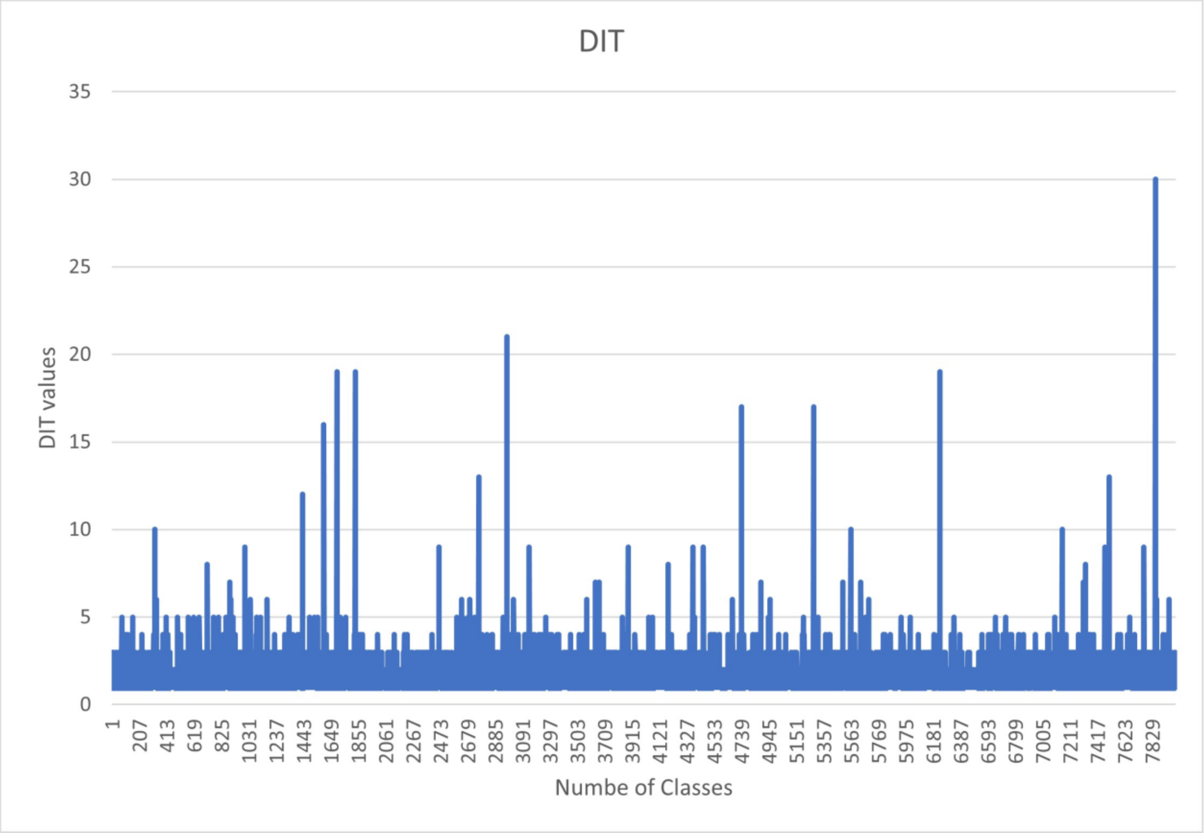
Hence The larger NOM the greater the impact on children – In order to achieve Understandability, Maintainability and Reusability the optimal value for the WMC should be **<50**

DIT

Maximum length from the node to the root- The number of methods inherited in a Class is 30

Minimum number of methods inherited a class is 1

Hence The larger NOM inherited high means higher reusability but higher complexity– In order to achieve Efficiency, Reusability, Understandability and Testability the optimal value for the **DIT** should be **<=5**

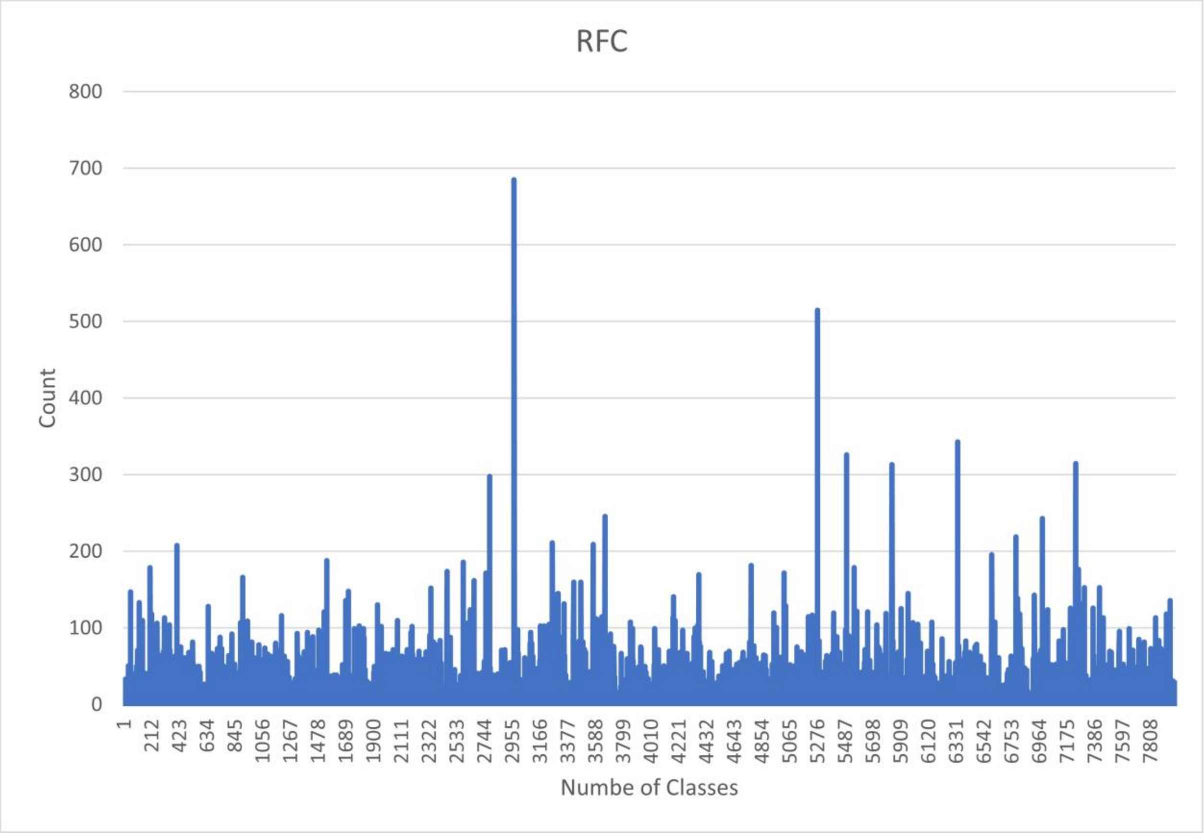


RFC

Maximum The number of methods can be invoked in response to a message to an object of the class is 685

Minimum number of methods: 0

Hence RFC is high means Increased complexity- testing and maintaining the class is harder. In order to achieve testability, maintainability and understandability the optimal value for the **RFC** should be **<=43.84**

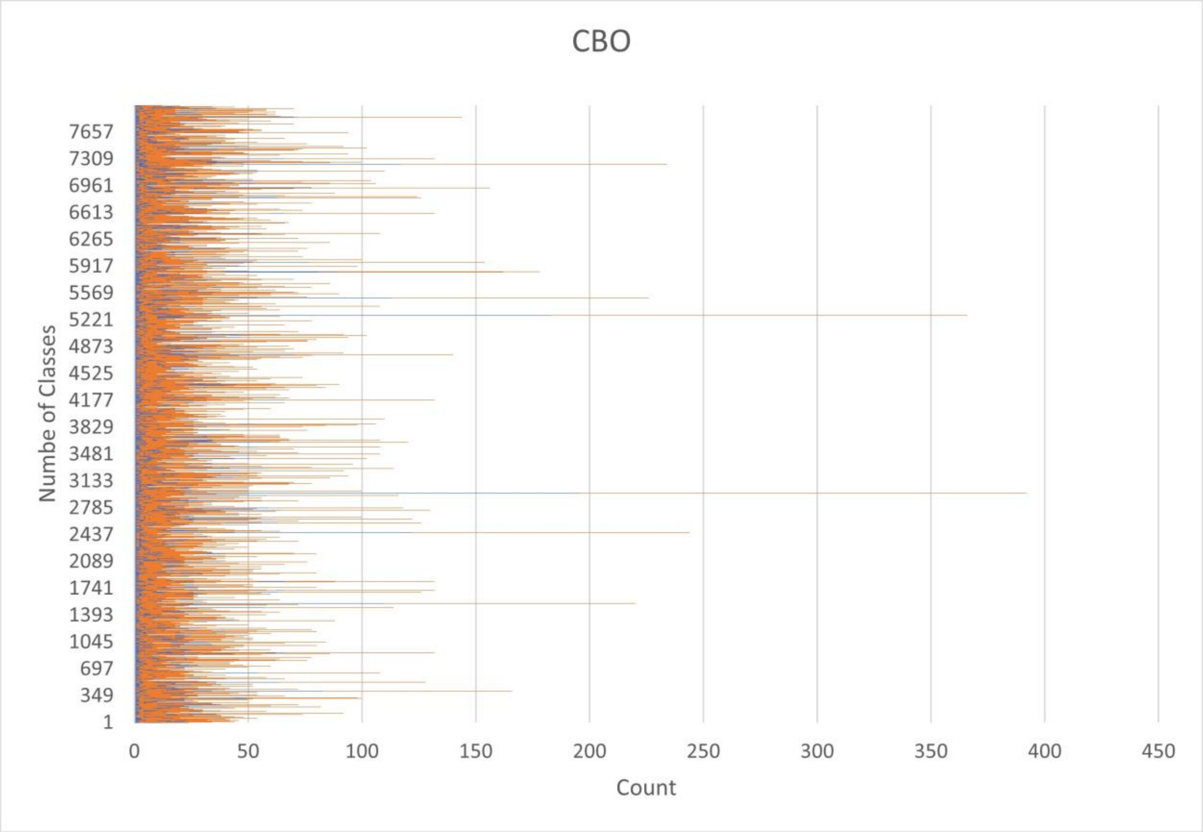


CBO

Maximum number of other classes to which a class is coupled is 196

Minimum is 0

Hence, Low CBO value Improve modularity which promote encapsulation, improves reusability, maintainability, and testability. So the optimal value for the **CBO** should be **<=14**

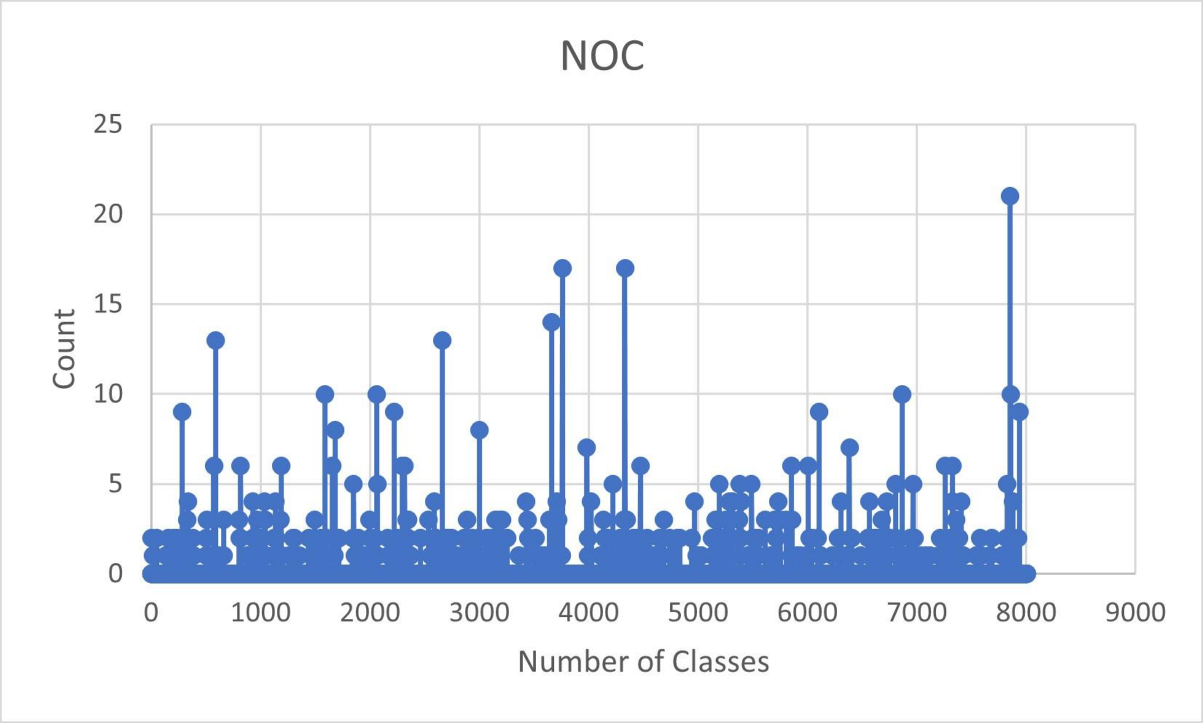


NOC

Maximum immediate subclasses subordinate to a class in the hierarchy 21

Minimum is 0

If a class has a large number of children, it may require more testing of the methods of that class, thus increase the testing time. Hence, the optimal value for the **NOC** should not be high.

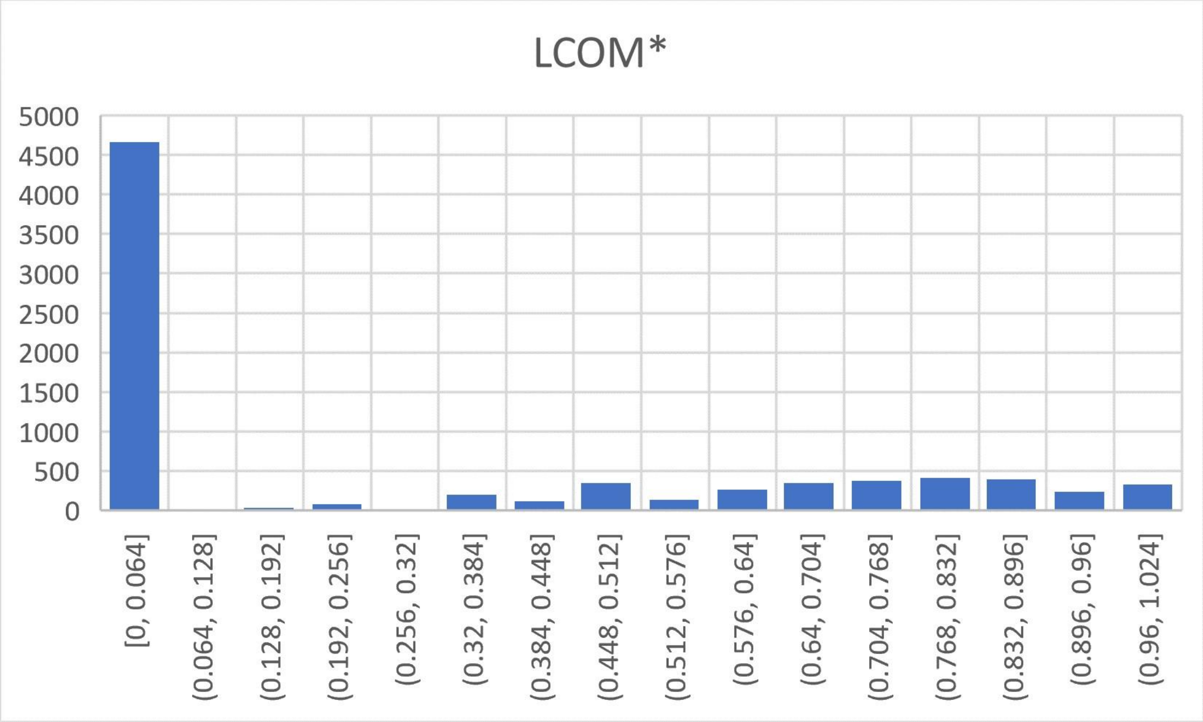


LCOM

Maximum dissimilarity of methods in a class by instance variable or attributes.is 1

Minimum number of methods: 0

Hence, Low value of LCOM Improves modularity which promote encapsulation, improves reusability, maintainability, and testability. The optimal value for the LCOM should be <=78.84.



Similarly, For project ***"Generate diagrams from textual description"***

Developing the Set of Goals: Improve the maintainability and design quality of the 'Generate diagrams from textual description' through the analysis of Chidamber and Kemerer (C&K) metrics.

Questions That Define the Goals in a Quantifiable Way:

1. What are the C&K metric values for classes in the 'Generate diagrams from textual description project?

2. Which classes exhibit the highest cyclomatic complexity (WMC) in the project?

3. Are there classes with deep inheritance hierarchies (DIT) that may need refactoring?

4. How does the number of children (NOC) vary across classes in the project?

5. Do classes with high coupling (CBO) pose maintainability challenges?

6. Which classes have a significant response for a class (RFC) value?

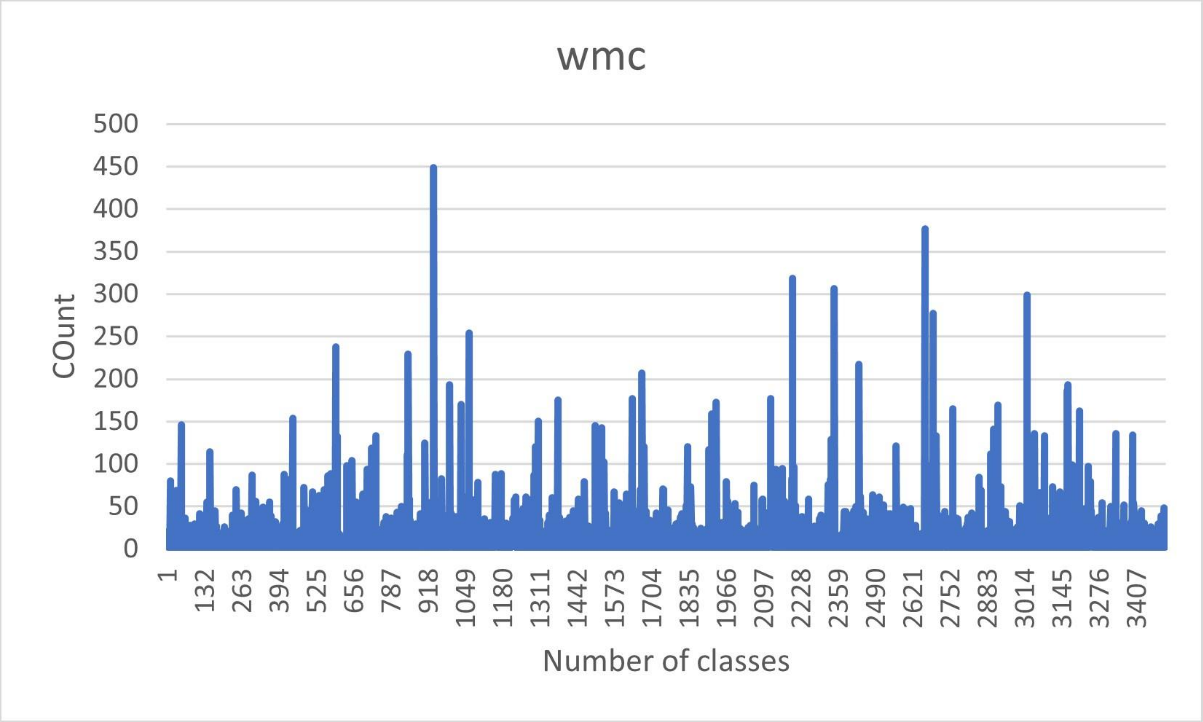
7. Is there a lack of cohesion in methods (LCOM) in any classes?

Specific Measures based on the Questions:

By using C&K Metrics tool, Collected the metrics information for the classes.

WMC: Maximum number of methods for in a Class is 449

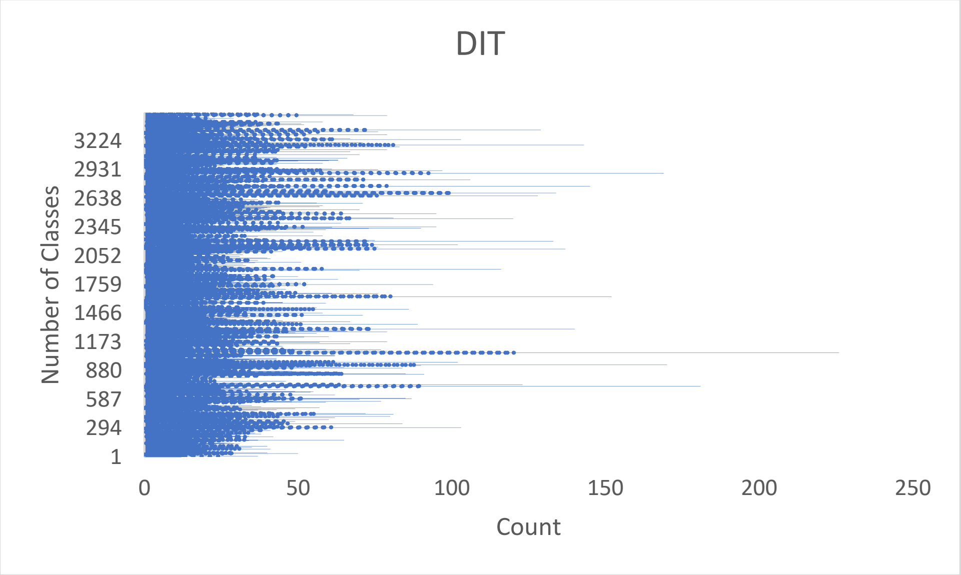
Minimum number of methods in a class is 0 → optimal value for the WMC should be **<50**



DIT

Maximum length from the node to the root- The number of methods inherited in a Class is 19

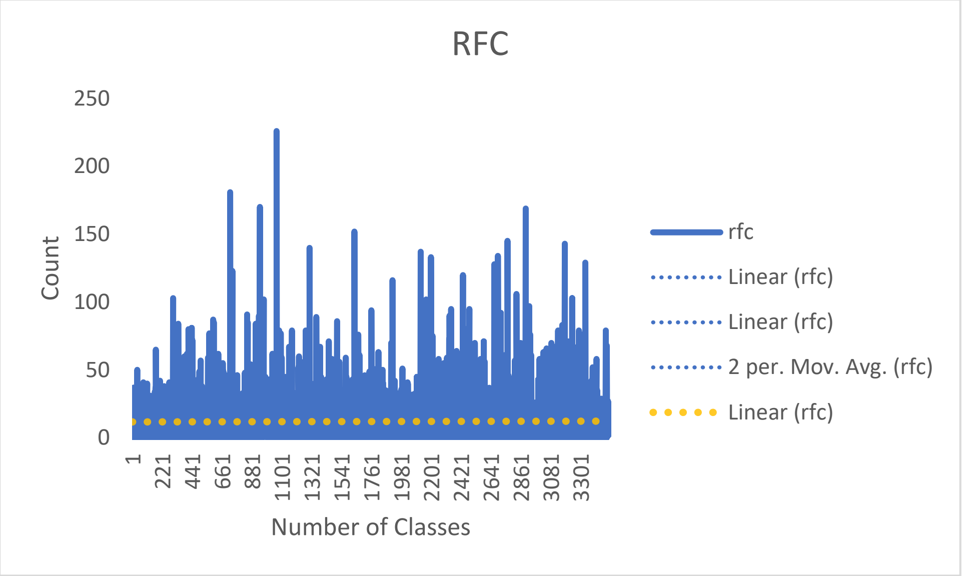
Minimum number of methods inherited a class is 1



RFC

Maximum The number of methods can be invoked in response to a message to an object of the class is 226

Minimum number of methods: 0

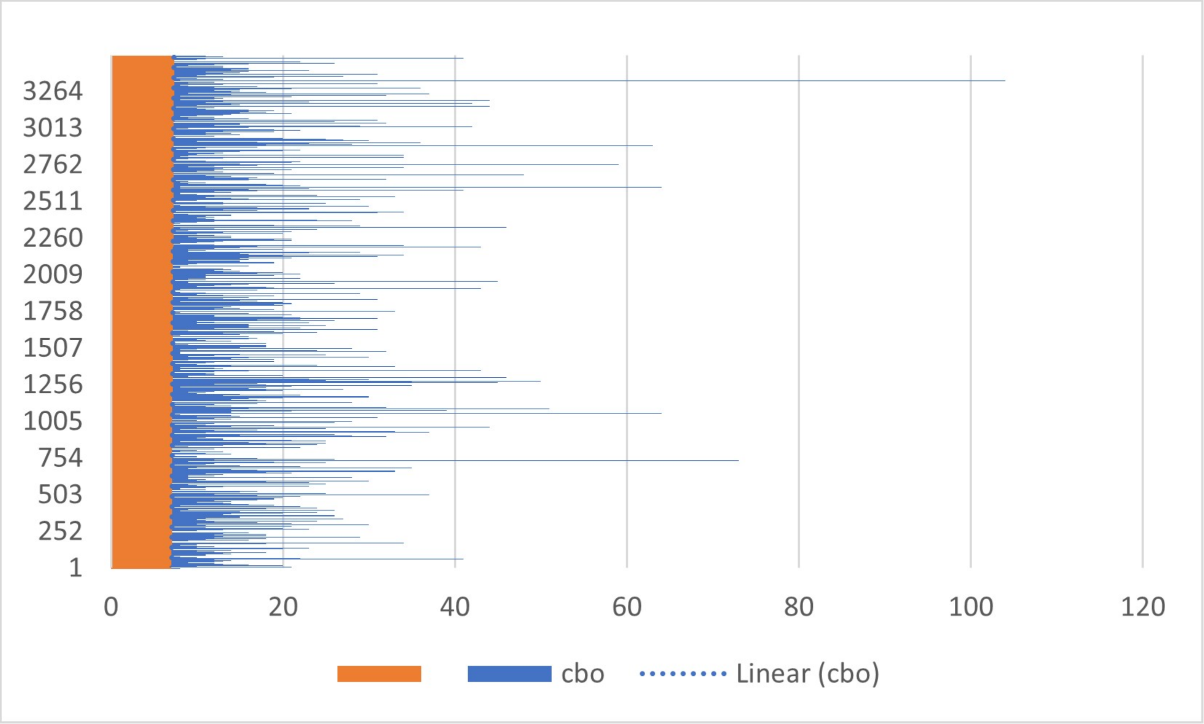


CBO

Maximum number of other classes to which a class is coupled is 104

Minimum is 0

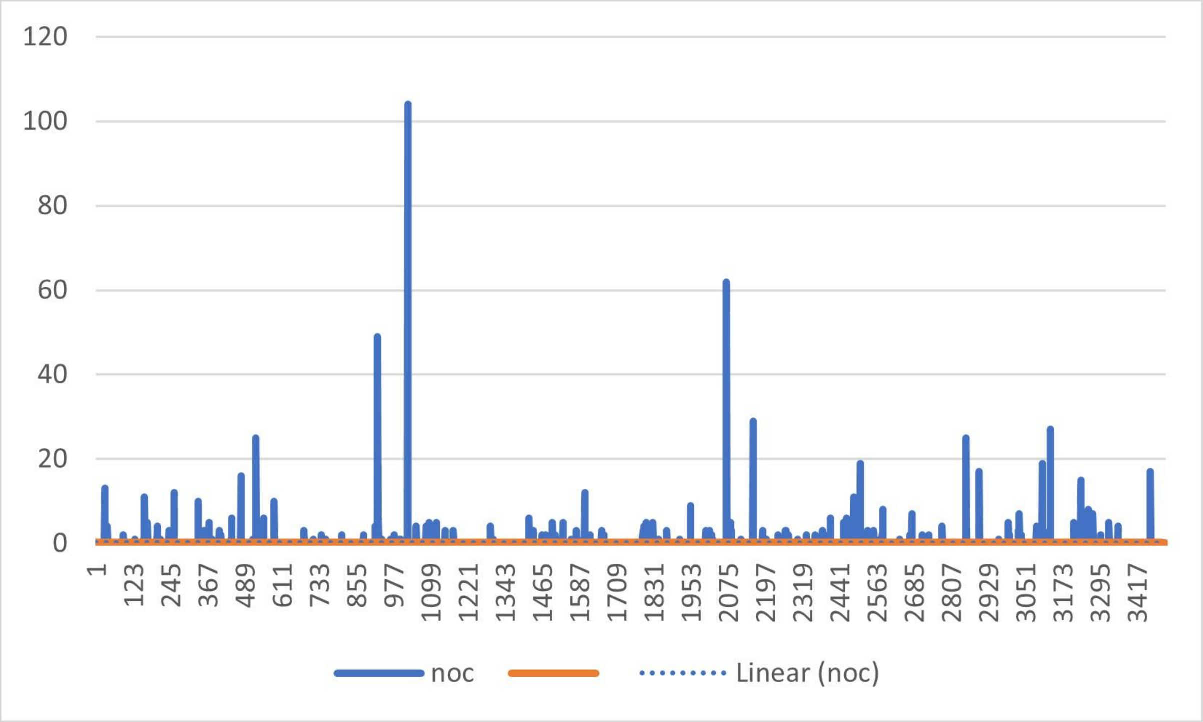
Hence, Low CBO value Improve modularity which promote encapsulation, improves reusability, maintainability, and testability. So the optimal value for the **CBO** should be **<=14**



NOC

Maximum immediate subclasses subordinate to a class in the hierarchy 104

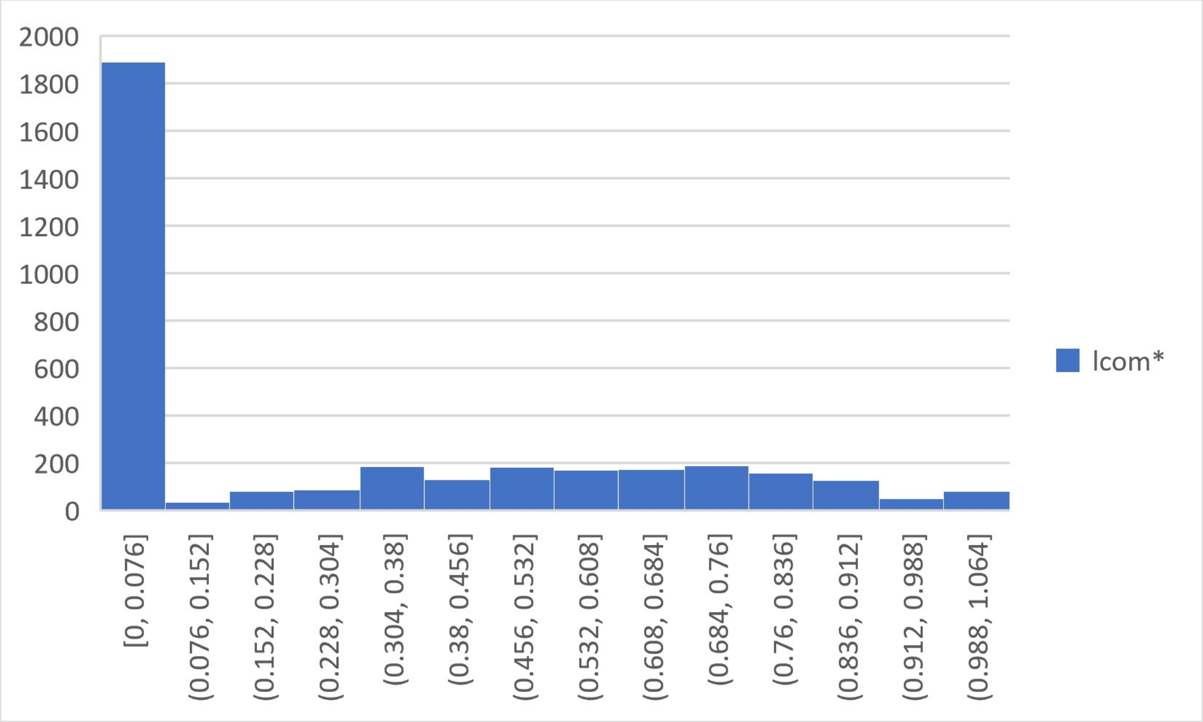
Minimum is 0



LCOM

Maximum dissimilarity of methods in a class by instance variable or attributes.is 1

Minimum number of methods: 0



By using C&K Metrics tools calculated the Metrics for the rest of the selected 3 programs.

Here are the Metrics with Min and Max Data points for all the Projects:

WMC Metric

Project	**Metric**	**Max**	**Min**	**Median**
DBeaver	WMC	920	0	4
Generate Diagrams	WMC	449	0	5
Grpahs for Everyone	WMC	495	0	4
OpenPDF	WMC	764	0	5
SpringData_MongoDB	WMC	380	0	3

DBeaver has a maximum WMC of 920, indicating that it contains classes with a high number of methods. While a high WMC can be a sign of complexity, DBeaver's median WMC is relatively low at 4. This suggests that while some classes may be complex, many classes have a manageable number of methods.

- The wide range (0 to 920) in WMC values indicates significant variation in class complexity, which can impact maintainability. Extremely complex classes may pose maintenance challenges, while simpler classes may be easier to maintain.

- The effect of class size on maintainability in DBeaver depends on the distribution of complex and simple classes. If the majority of classes have a low or moderate WMC, the impact on maintainability may be manageable.

Generate_Diagrams has a maximum WMC of 449, with a median of 5. This indicates that the majority of classes in this project are relatively small in terms of the number of methods.

- The low median suggests that most classes have a limited number of methods, which can contribute to better maintainability.

- In this case, the effect of class size on maintainability is likely positive, as smaller classes are typically easier to understand, modify and maintain.

Graphs for Everyone has a maximum WMC of 495, with a median of 4. Similar to Generate_Diagrams, the majority of classes have a relatively low number of methods.

- The low median value suggests that the project prioritizes smaller and potentially more maintainable classes.

- The effect of class size on maintainability is likely positive, with a focus on smaller, more manageable classes

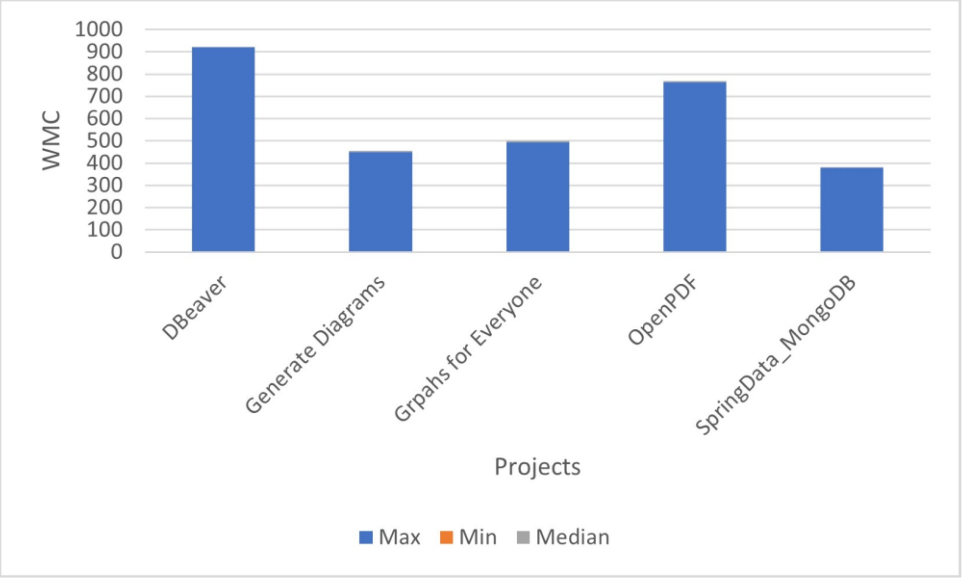
OpenPDF has a maximum WMC of 764, with a median of 5. Similar to DBeaver, OpenPDF exhibits a wide range in WMC values, with a significant number of methods in some classes.

- The project may have some complex classes that could pose maintainability challenges, especially if these complex classes are critical to the system's functionality.

- The effect of class size on maintainability in OpenPDF is mixed, with both small and complex classes, suggesting the need to carefully manage and refactor highly complex classes.

- ***SpringData_MongoDB*** has a maximum WMC of 380, with a median of 3. This project places a stronger emphasis on smaller classes, with a median WMC of only 3.

- The effect of class size on maintainability in SpringData_MongoDB is likely positive due to its focus on smaller, more concise classes, which are generally easier to maintain.



DIT Metric

Project	Metric	Max	Min	Median
Dbeaver	DIT	30	1	1
Generate Diagrams	DIT	19	1	1
Grpahs for Everyone	DIT	58	1	1
OpenPDF	DIT	17	1	1
SpringData_MongoDB	DIT	48	1	1

Analysing the Depth of Inheritance Tree (DIT) metrics for the studied projects, which measures the depth of the inheritance hierarchy, we can draw the following observations regarding the potential effect of class hierarchy depth on maintainability.

DBeaver has a maximum DIT of 30, indicating that some classes have a relatively deep inheritance hierarchy. The minimum DIT is 1, suggesting that there are classes with no inheritance relationships. The median DIT is also 1, indicating that the majority of classes have a shallow inheritance hierarchy.

- In DBeaver, the overall effect of class hierarchy depth on maintainability seems to vary. While many classes have a shallow hierarchy, the presence of classes with deeper hierarchies may introduce complexities that need to be managed.

Generate Diagrams has a maximum DIT of 19, with a minimum and median DIT of 1. This suggests that the majority of classes have a shallow inheritance hierarchy.

- The overall effect of class hierarchy depth on maintainability in Generate Diagrams appears to be positive due to the prevalence of shallow hierarchies, which are typically easier to understand and maintain.

Graphs for Everyone has a maximum DIT of 58, with a minimum and median DIT of 1. Similar to Generate Diagrams, the project predominantly features classes with shallow hierarchies.

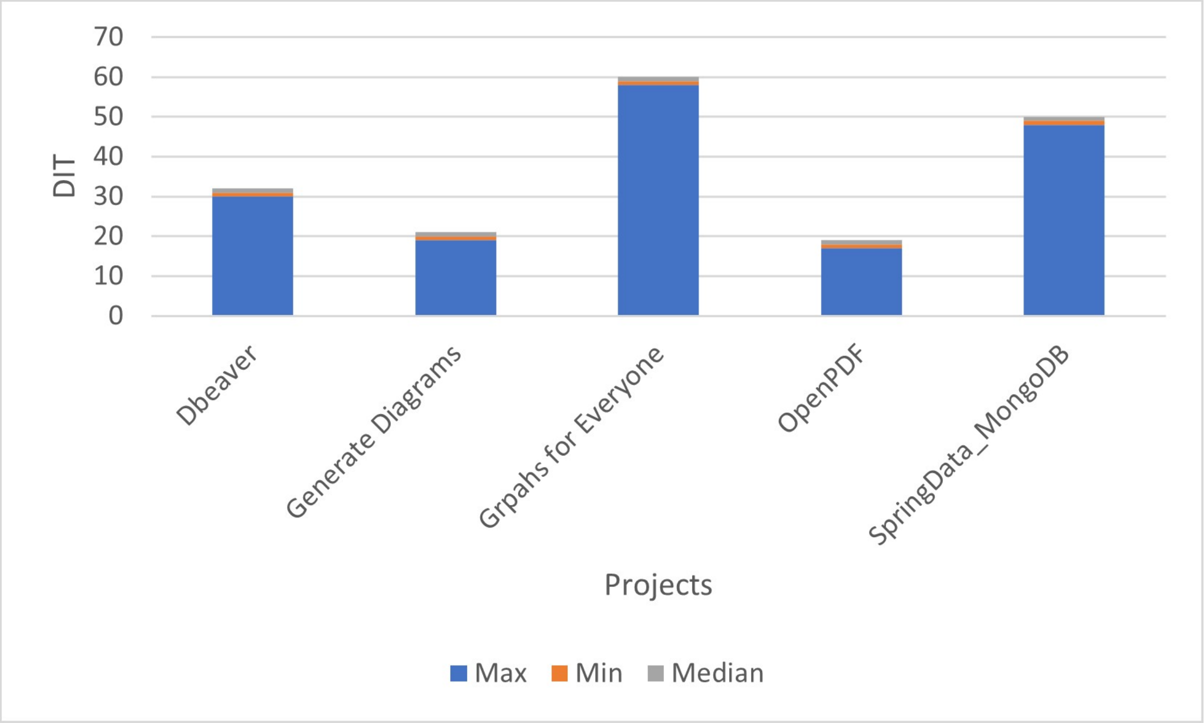
- The overall effect of class hierarchy depth on maintainability in Graphs for Everyone is likely positive, as the emphasis on shallow hierarchies simplifies the design and maintenance.

OpenPDF has a maximum DIT of 17, with a minimum and median DIT of 1. This project, like Generate Diagrams and Graphs for Everyone, emphasizes shallow hierarchies.

- The overall effect of class hierarchy depth on maintainability in OpenPDF is likely positive, as it follows a design pattern that minimizes complex inheritance.

SpringData_MongoDB has a maximum DIT of 48, with a minimum and median DIT of 1. The project primarily features classes with shallow hierarchies.

- The overall effect of class hierarchy depth on maintainability in SpringData_MongoDB is positive due to the prevalence of shallow hierarchies, which simplifies the structure



RFC Metric

Project	Metric	Max	Min	Median
Dbeaver	RFC	685	0	4
Generate Diagrams	RFC	226	0	6
Grpahs for Everyone	RFC	287	0	3
OpenPDF	RFC	424	0	10
SpringData_MongoDB	RFC	504	0	2

High RFC values may indicate complexity and potentially impact maintainability, the presence of classes with a low RFC positively influences maintainability. The overall impact on maintainability depends on the distribution of classes with varying RFC values and how effectively the project manages and refactors complex classes to improve their maintainability.

DBeaver has a wide range of RFC values, from 0 to 685, with a median RFC of 4. The high maximum RFC value indicates that some classes in DBeaver have a large number of methods that can be executed in response to messages. The median RFC value is relatively low, suggesting that a significant number of classes have a manageable response for a class.

- The effect of class size on maintainability in DBeaver is mixed. While some classes may have a high RFC, many classes appear to have a moderate or low RFC, which can positively impact maintainability.

Generate Diagrams has a wide range of RFC values, with a maximum RFC of 226 and a median RFC of 6. The variation in RFC values suggests that some classes have a relatively high number of methods.

- The effect of class size on maintainability in Generate Diagrams seems mixed. While some classes may have higher complexity, others have a more manageable response for a class, potentially enhancing maintainability.

Graphs for Everyone project has a maximum RFC of 287 and a median RFC of 3. The median RFC is relatively low, indicating that many classes have a low response for a class.

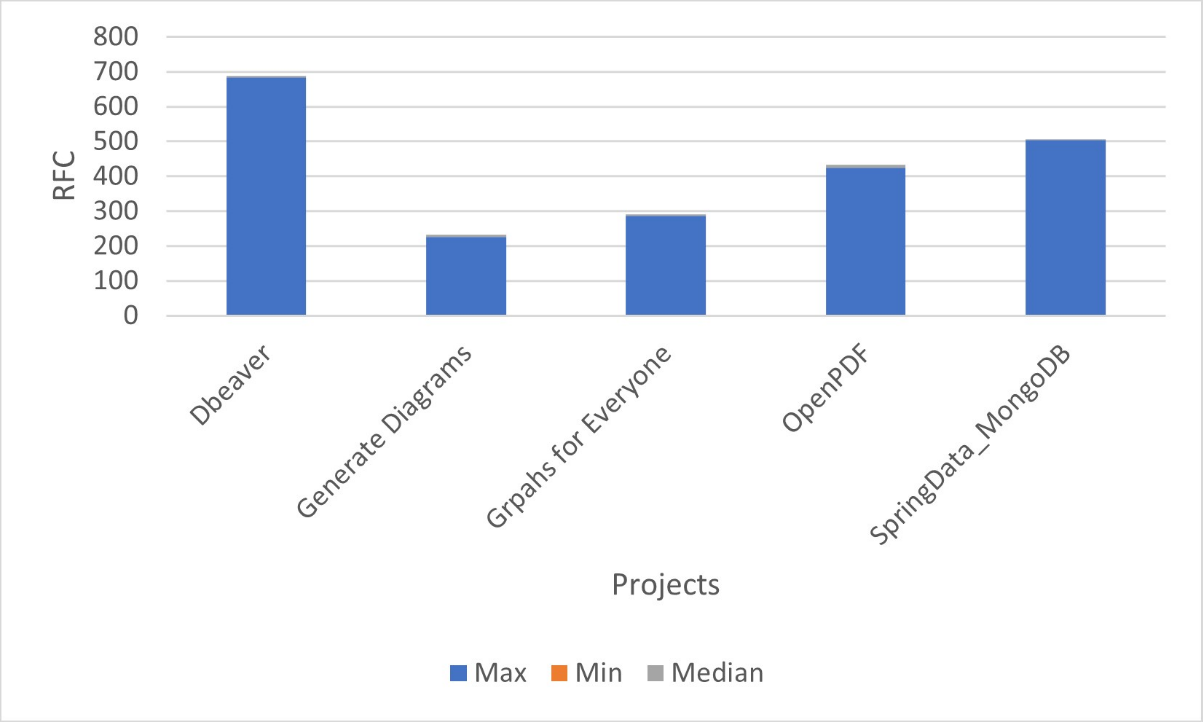
- The effect of class size on maintainability in Graphs for Everyone appears to be positive. The prevalence of classes with a low RFC suggests a focus on maintainability through simpler class structures.

OpenPDF has a maximum RFC of 424, and the median RFC is 10. The variation in RFC values suggests that some classes in OpenPDF have a high response for a class.

- The effect of class size on maintainability in OpenPDF is mixed. While some classes may have high complexity, others have a low RFC, which can contribute to better maintainability.

SpringData_MongoDB exhibits a wide range of RFC values, with a maximum RFC of 504 and a median RFC of 2. The median RFC is low, indicating that many classes have a low response for a class.

- The effect of class size on maintainability in SpringData_MongoDB seems positive. The focus on classes with a low RFC suggests an emphasis on maintainability.



NOC Metric

Project	Metric	Max	Min	Median
Dbeaver	NOC	21	0	0
Generate Diagrams	NOC	104	0	0
Grpahs for Everyone	NOC	43	0	0
OpenPDF	NOC	35	0	0
SpringData_MongoDB	NOC	242	0	0

Analyzing the Number of Children (NOC) metrics for the studied projects, which measures the number of immediate subclasses a class has. In projects where classes do not have immediate subclasses (NOC=0), the effect on maintainability is likely positive, as it simplifies class relationships. However, in projects with some classes having a high number of immediate subclasses, complexity may be introduced, potentially impacting maintainability.

DBeaver has a maximum NOC of 21, indicating that some classes have a relatively high number of immediate subclasses. The minimum NOC is 0, which implies that some classes have no immediate subclasses. The median NOC is 0, suggesting that the majority of classes do not have immediate subclasses.

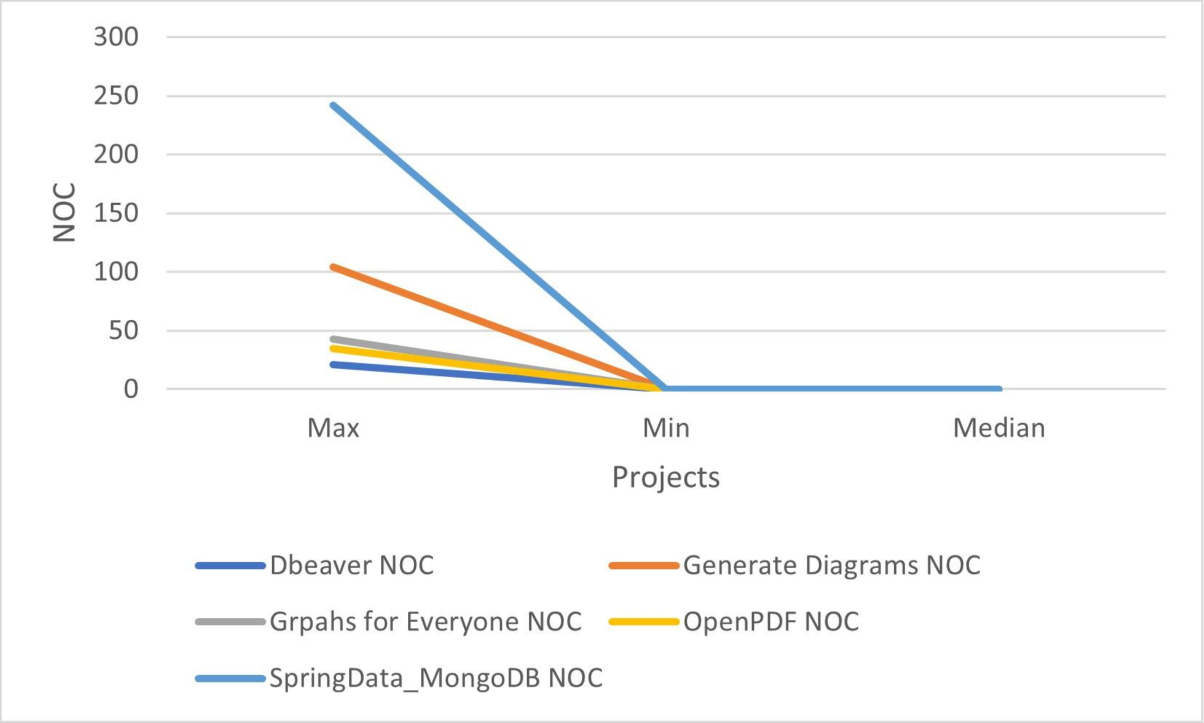
Generate Diagrams has a maximum NOC of 104, with a minimum NOC of 0 and a median NOC of 0. This suggests that some classes may have a relatively high number of immediate subclasses.

Graphs for Everyone has a maximum NOC of 43 and a median NOC of 0. The majority of classes do not have immediate subclasses. The effect of class size on maintainability in Graphs for Everyone seems positive.

OpenPDF has a maximum NOC of 35, with a median NOC of 0. The majority of classes do not have immediate subclasses. The effect of class size on maintainability in OpenPDF also appears positive. The absence of immediate subclasses simplifies class relationships and can improve maintainability.

SpringData_MongoDB exhibits a maximum NOC of 242 and a median NOC of 0. Many classes have a high number of immediate subclasses.

- The effect of class size on maintainability in SpringData_MongoDB is mixed. The presence of classes with a high NOC may introduce complexity, but classes without immediate subclasses may have better maintainability.



CBO Metric

Project	Metric	Max	Min	Median
Dbeaver	CBO	196	0	4
Generate Diagrams	CBO	104	0	5
Grpahs for Everyone	CBO	155	0	4
OpenPDF	CBO	84	0	4
SpringData_MongoDB	CBO	177	0	3

Analysing the Coupling Between Objects (CBO) metrics for the studied projects, which measures the number of other classes to which a class is coupled. While some classes may have a high level of coupling (CBO) with other classes, the majority of classes exhibit a moderate to low level of coupling.

DBeaver has a maximum CBO of 196, indicating that some classes are highly coupled with a large number of other classes. The minimum CBO is 0, which implies that some classes are not coupled with any other classes. The median CBO is 4, on average, classes have a moderate level of coupling with other classes.

- The effect of class size on maintainability in DBeaver is mixed. While some classes have a high level of coupling, the median CBO value indicates that many classes have a moderate level of coupling, which can positively impact maintainability.

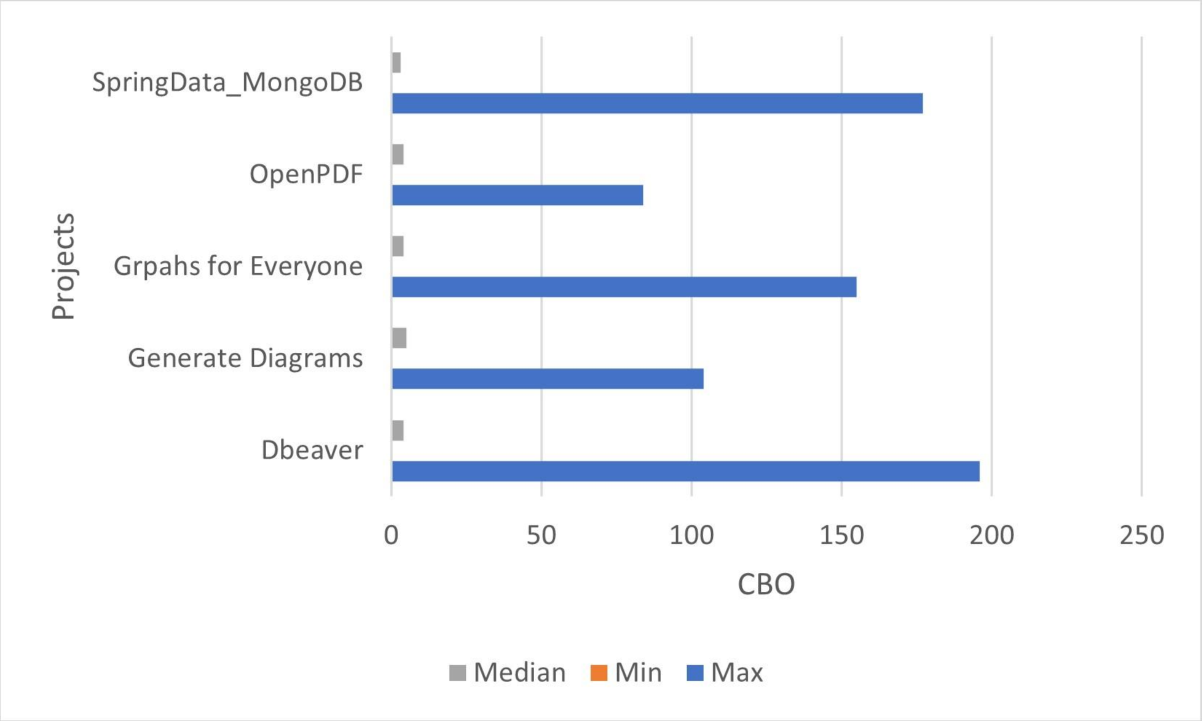
Generate Diagrams has a maximum CBO of 104, with a minimum CBO of 0 and a median CBO of 5. This suggests that some classes have a moderate level of coupling with other classes. The effect of class size on maintainability in Generate Diagrams seems mixed.

Graphs for Everyone has a maximum CBO of 155, with a minimum CBO of 0 and a median CBO of 4. This indicates that many classes have a moderate level of coupling with other classes. The effect of class size on maintainability in Graphs for Everyone appears mixed.

OpenPDF has a maximum CBO of 84, with a minimum CBO of 0 and a median CBO of 4. The majority of classes have a moderate level of coupling.

- The effect of class size on maintainability in OpenPDF is high. Classes with a moderate level of coupling may have challenges related to maintainability, but those without coupling may be more maintainable.

SpringData_MongoDB exhibits a maximum CBO of 177 and a median CBO of 3. This suggests that some classes are coupled with a relatively high number of other classes, while others have lower coupling.



References

- D. Baldwin, F. Sayward, Heuristics for determining equivalence of program mutations, Technical Report Research Report...
- N. Fenton *et al.*
  Software Metrics: A Rigorous and Practical Approach
  (1999)
- Basili, V. R.; Weiss, D. M. (November 1984). "A Methodology for Collecting Valid Software Engineering Data". *IEEE Transactions on Software Engineering*. SE-10

(6): 728–738. doi:10.1109/TSE.1984.5010301. hdl:1903/7513. ISSN 1939-3520. S2CID 10114944.

- W. Boehm, J. R. Brown, and M. Lipow, "Quantitative Evaluation of Software Quality," Proceedings of the Second International Conference on Software Engineering, 1976, pp.592-605.
- V.R. Basili, "Quantitative Evaluation of Software Engineering Methodology," Proceedings of the First Pan Pacific Computer Conference, Melbourne, Australia, September 1985.
- [Chidamber and Kemerer, 1994] S. R. Chidamber and C. F. Kemerer, "A metric suite for object oriented design," IEEE Transactions on Software Engineering, pp. 476–493, 1994.
- [Fenton and Pfleeger, 1997] Norman E. Fenton, Shari Lawrence Pfleeger. Software Metrics. A rigorous & Practical Approach. 2nd Edition. ITP, International Thomson Computer Press, 1997
- https://sites.pitt.edu/~ckemerer/CK%20research%20papers/MetricForOOD_ChidamberKemerer94.pdf
- R. Chillarege, I. S. Bhandari, J. K. Chaar, M. J. Halliday, D. S. Moebus, B. K. Ray, and M. Y. Wang. On the value of static analysis for fault detection in software measurements. IEEE Trans. Software Engineering, 18(11):943-956, Nov 1992.