# Metrics #1 - Dependency Metrics

**Author:**

Team member 56971, João Vieira

## Introduction

Utilizing the metrics reloaded plugin in IntelliJ IDEA, we can obtain metrics on the number of many different types of dependencies.

All this metrics gives us information that allows us to identify potential trouble spots in the codebase. However, this is not the case for this project, as the values obtained in the metrics are within the plugin’s thresholds, therefore the values of this metrics are not enough to identify potential code smells and other problems, as we will see further in this report.

To simplify the reading process, the following abbreviations to the various dependencies were used:

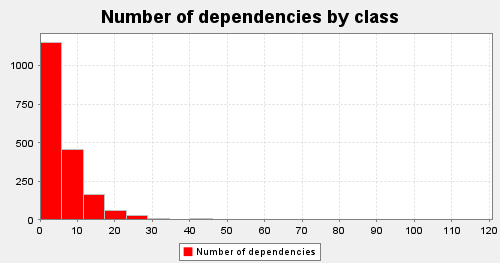
* Cyclic – Number of cyclic dependencies
* Dcy – Number of dependencies
* PDpt – Number of dependent packages
* Dpt – Number of dependents
* PDcy – Number of package dependencies
* Dcy\*– Number of transitive dependencies
* Dpt\*– Number of transitive dependents

## Class Metrics

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Cyclic | Dcy | Dcy\* | Dpt | Dpt\* | PDcy | PDpt |
| Min | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Max | 784 | 115 | 1398 | 591 | 1358 | 52 | 103 |
| Avg | 310,8336 | 6,049285 | 765,8389 | 5,225225 | 751,0705 | 3,705882 | 2,503445 |

As I said in the beginning none of the values exceed the thresholds set by the plugin, but those thresholds aren’t necessarily relevant to our project, so we utilized boxplots and histograms to get a more accurate representation of our project. Of the seven boxplots, the following stood out.

These two boxplot graphs tell us that although most classes have their number of transitive dependencies (left) and transitive dependents (right) within the thresholds of the plugin and near the average value, there is a small number of classes that stand out and have some extreme values. Transitive dependencies aren’t usually problem as most of them can be solved by the compiler. That being said, many transitive dependencies could lead to the creation of cyclic dependencies and potentially cause many unwanted negative effects.



There are a few classes that have some extreme number of dependencies, being one them the org.jabref.gui.JabRefFrame class that has the highest amount of dependencies with 113. Looking at the code we can see some methods that show signs of the code smell Inappropriate Intimacy:

Uma imagem com texto

Descrição gerada automaticamente

Uma imagem com mesa

Descrição gerada automaticamente

In the case of the number of dependents we are able to observe that there are some classes with extreme values.

The org.jabref.model.entry.BibEntry with 591 number of dependents, almost one hundred times bigger than the average, is a very crucial class, that must be altered very carefully, as any alteration could cause a lot of classes from the project to break.

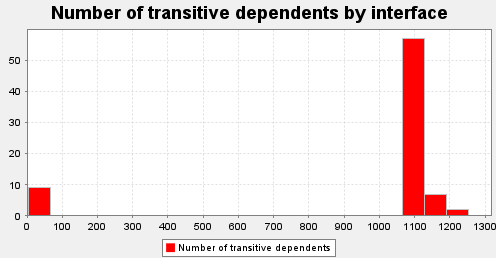
## Interface Metrics

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Cyclic | Dcy | Dcy\* | Dpt | Dpt\* | PDcy | PDpt |
| Min | 0 | 0 | 0 | 1 | 5 | 0 | 1 |
| Max | 784 | 56 | 1335 | 219 | 1254 | 30 | 67 |
| Avg | 386,7733 | 2,586667 | 747,3867 | 21,54667 | 993,52 | 1,813333 | 6,866667 |

When observing this histograms, theres some values that should concern us.

Uma imagem com mesa

Descrição gerada automaticamente



Most of our interfaces have their number of transitive dependents with values between 1121 and 1254. Which could lead to the creation of more cyclic dependencies which can cause the involving classes and interfaces to be dependent on one another, making it very hard to develop and refactor the code in the future, as one change in one class or interface would affect many and possibly lead to many unwanted negative effects.

## Package Metrics

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Cyclic | PDcy | PDpt | PDpt\* |
| Min | 0 | 0 | 0 | 1 |
| Max | 153 | 86 | 102 | 166 |
| Avg | 106,1351 | 8,234234 | 8,234234 | 147,5348 |

Uma imagem com mesa

Descrição gerada automaticamente

The following two graphs stand out because, transitive dependencies seem to be a recurring issue in the project. Like I said before, transitive dependencies can lead to creation of cyclic dependencies, making it harder to develop and refactor the code in the future.

## Conclusion

In conclusion, our project is acceptable and reasonable state when it comes to dependencies. However future development of code must take into consideration these metrics, as all of them showed a very high number of cyclic and transitive dependencies, which, has I said before, can be problematic since they can lead to more cyclic dependencies that can result in unwanted negative effects when trying to refactor the code.