**Class dependency metric**

This metric shows the number of dependencies related to classes, as it is possible to know information about how many classes/interfaces depend on a certain class, how many modules is each class dependent on, and some more information.

With this metric, we can see several values:

* Number of cyclic dependencies (***Cyclic***) – number of modules where the dependency goes the two ways regarding each class, e.g. if we are evaluating a class A, the *Cyclic* calculates the number of classes that depend on A and that A is dependent on, simultaneously.
* Number of dependencies (***Dcy***) – number of classes which each class directly depends on.
* Number of transitive dependencies (***Dcy\****) – number of classes which each class directly or indirectly depends on.
* Number of dependents (***Dpt***) – number of classes that directly depend on each class.
* Number of transitive dependents (***Dpt\****) – number of classes that directly or indirectly depend on each class.
* Number of package dependencies (***PDcy***) – number of packages which each class directly depends on.
* Number of dependent packages (***PDpt***) – number of packages that directly depend on each class.

Generally, a high number of class dependencies means more coupling, which is the degree each module depends on the others. It is used to measure the strength of associations between different modules, so the higher the coupling is the more complex the system tends to be. This can lead to several problems, making the project much more difficult to debug and to eventually extend it, by adding more functionalities, for instance. If the calculated dependency numbers of a class is too high in comparison to the average, that possibly means that the coupling is also elevated, which can translate into an unnecessary increase of complexity in the overall project.

Looking at the retrieved statistics from the IntelliJ plugin, there are a few values that seem to be above the average.

The ***Cyclic*** of each class fluctuates between two values, 0 and 515, and the second value seems a bit above the normal range, which can mean that there are too many cycles. This two-way dependency can be an issue in a way that it makes the code harder to understand and extend, as well as debug.

The ***Dcy*** had only one value that was a little extreme, in the GanttProject class (246), being this a possible troublespot. This can be explained by the fact that this is the main frame of the project, and it’s necessary that this class communicates with several other classes, so it’s understandable that the GanttProject has more dependencies than the remaining classes.

***Dcy\**** and ***Dpt\**** has a value that is shown many times, 867 and 560, respectively. This value is not particularly low, which can be explained by the fact that many classes communicate with many classes indirectly, being that the difference between transitive and normal dependency. This is a possible troublespot, as the level of coupling is too high, and that may difficult the understanding of the code.

The ***Dpt*** is significantly above the others in three classes: GanttLanguage(116) , GPAction(97) and GPLogger(81). GanttLanguage is the class responsible for the language of the app, dealing with some formats of dates, which are abundantly used throughout the project, mainly regarding tasks, being an explanation for this high value. GPAction is the class responsible for the actions (such as *ok* and *cancel*) associated with events. Lots of other classes take care of the consequences of these actions, so this can be a possible reason. *GPLogger* is responsible for the ‘Log’ in the ‘Help’ tab, which shows errors, warnings, etc., so it makes sense to have many classes dependent on it, since the things shown in the log are done in several different classes.

The ***PDcy*** is relatively high in GanttProject possibly because, as we said in ***Dcy***, this is the main frame of the project, so it depends on a lot of classes, from different packages, as it takes care of lots of different information.

The ***PDpt*** has high values in GanttLanguage(31) and GPLogger(27). The explanation is similar to the one described in the ***Dpt*** paragraph.

There are a couple of code smells referred in the first phase that may impact these statistics: the data class, which is a class with information only (getters), may cause more unnecessary coupling; message chain, in which many classes depend on each other to call many methods from different classes successively, increasing the number of dependencies.

In conclusion, many of these values are extremely big, and they correspond mostly to complex classes that take care of important information used in most of the project, such as GanttProject. However, some values are enormous in almost every class, and by that we can conclude the project has an insanely big level of coupling, resulting in many code understanding and extending problems, which were strongly felt during the code study part of the project asssignment.

**Bibliography:**

SHARMA, Shweta et al./ International Journal of Computer Science & Engineering Technology (IJCSET)

https://www.techtarget.com/searchapparchitecture/tip/The-basics-of-software-coupling-metrics-and-concepts

SHAIKH, Mohsin, International Journal of Advanced Computer Science and Applications, Vol. 8, No. 10, 2017