Lipan Andreea-Alexandra
UBB FMI Software Engineering Master
20 March 2025

CodeMR analysis report on my bachelor thesis

My bachelor thesis application

It is an Android application, written in Kotlin. It implements an Android launcher and functionalities such as icons customisation, share screen with other app users, and remote control in app. Also uses speech to text API.

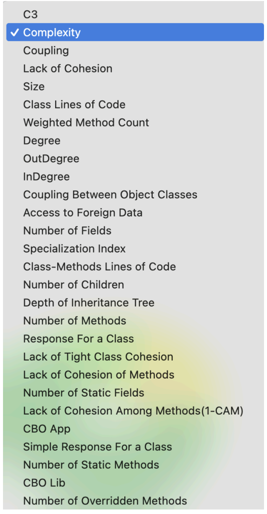
CodeMR

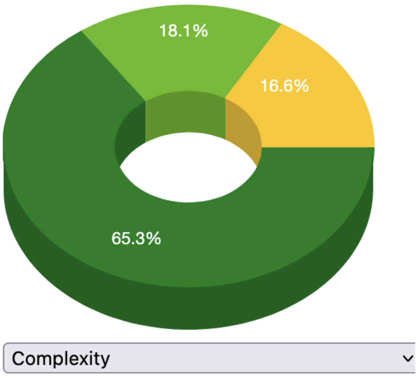
CodeMR is a software quality and static code analysis tool for Java, Kotlin and Scala projects. It visualizes code metrics and high level quality attributes (coupling, complexity, cohesion and size) in different views, such as Package Structure, TreeMap, Sunburst, Dependency and Graph Views [1].

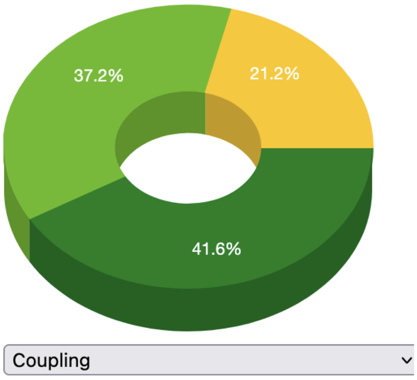
I've installed the plugin in Android Studio, selected my project module, clicked the CodeMR option in the menu and then it generated some local files with the analysis.

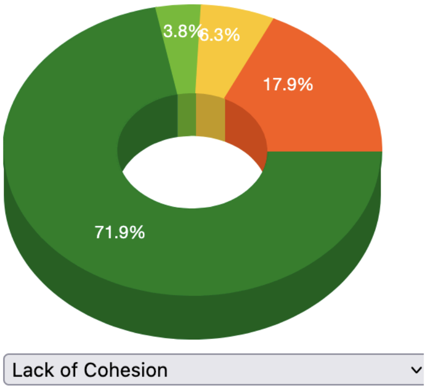
Results of using the tool & Interpretation

The tool has quite a few metrics. All metrics are explained in a separate page, but there is no way to hover over a name and get the explanation for what the metric means which is rather inconvenient. The explanations of the metrics are okay, you can kinda understand what it is about, but only a few of them say why a metric is important or what impact it might have, which I find silly. If you put a metric in your application you must tell the user why that's there and why should the user care about it. Seems like a basic functionality to me.

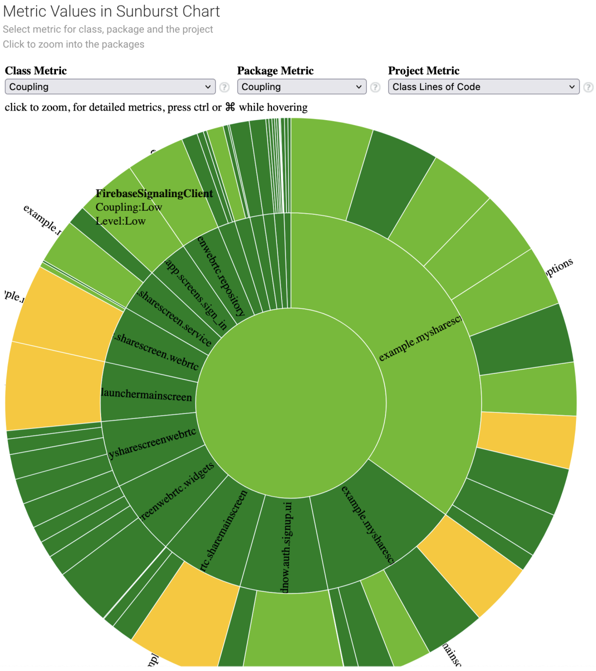








These donut diagrams are an overview of the most important metrics. The software also has lots of other diagrams. Some are more useful than others. The software lacks any sort of description of these diagrams, so you must figure it out on your own.



Metric Values in Sunburst Chart

Select metric for class, package and the project

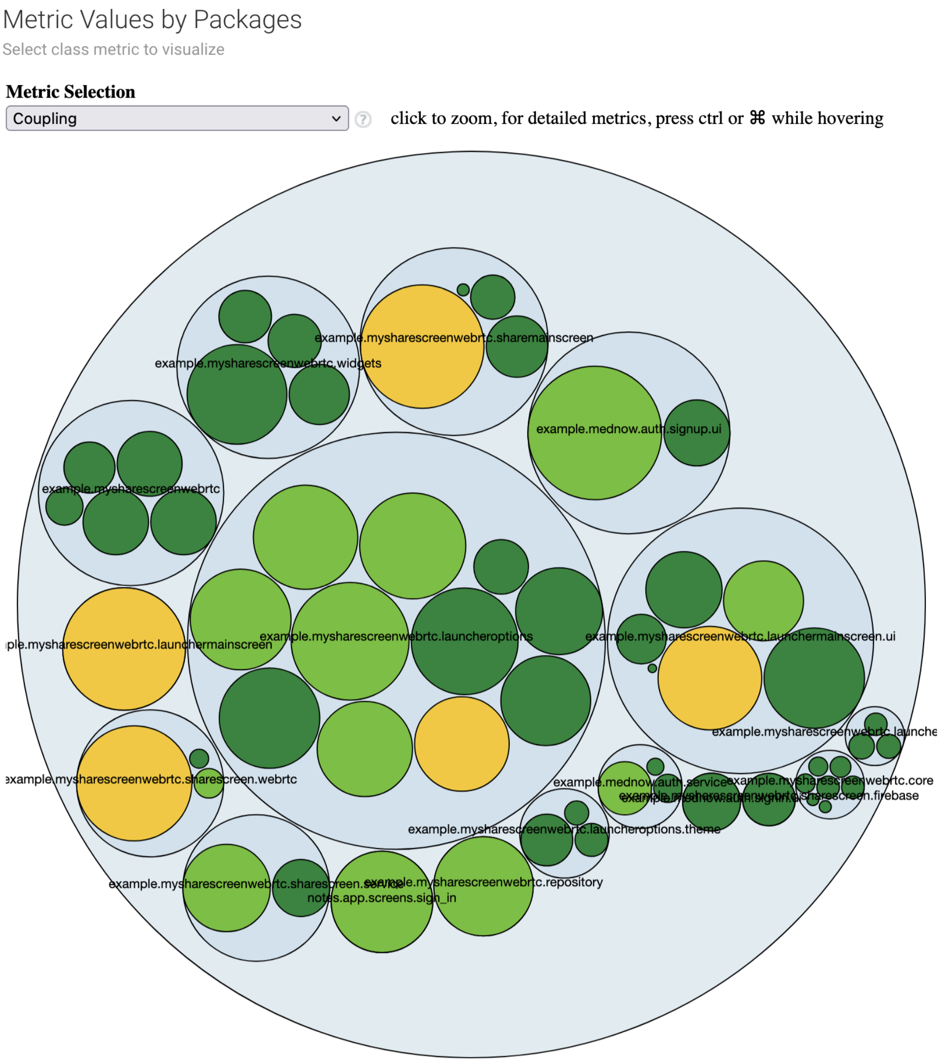
Click to zoom into the packages

Class Metric: Coupling

Package Metric: Coupling

Project Metric: Class Lines of Code

click to zoom, for detailed metrics, press ctrl or ⌘ while hovering



Metric Values by Packages

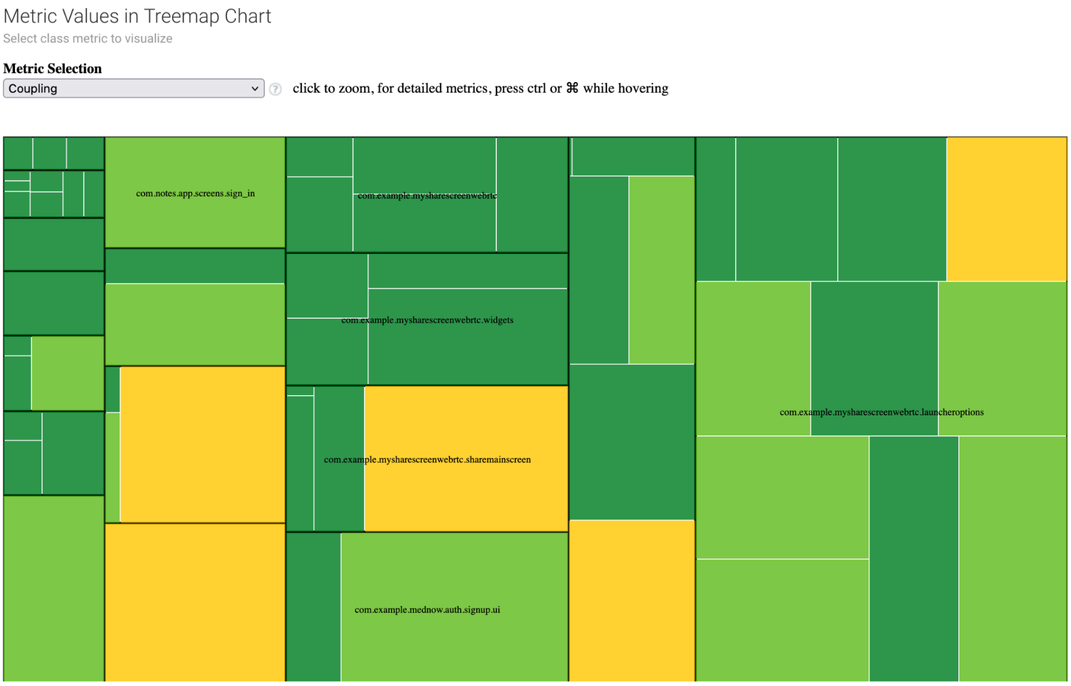
Select class metric to visualize

Metric Selection

Coupling

click to zoom, for detailed metrics, press ctrl or ⌘ while hovering

example.mysharescreenwebrtc.sharemainscreen
example.mysharescreenwebrtc.widgets
example.mednow.auth.signup.ui
example.mysharescreenwebrtc
ple.mysharescreenwebrtc.launchermainscreen
example.mysharescreenwebrtc.launcheroptions
example.mysharescreenwebrtc.launchermainscreen.ui
example.mysharescreenwebrtc.launche
xample.mysharescreenwebrtc.sharescreen.webrtc
example.mednow.auth.service
example.mysharescreenwebrtc.core
example.mysharescreenwebrtc.sharescreen.firebase
example.mednow.auth.signin.ui
example.mysharescreenwebrtc.launcheroptions.theme
example.mysharescreenwebrtc.sharescreen.service
example.mysharescreenwebrtc.repository
notes.app.screens.sign_in



Metric Values in Treemap Chart

Select class metric to visualize

Metric Selection

Coupling

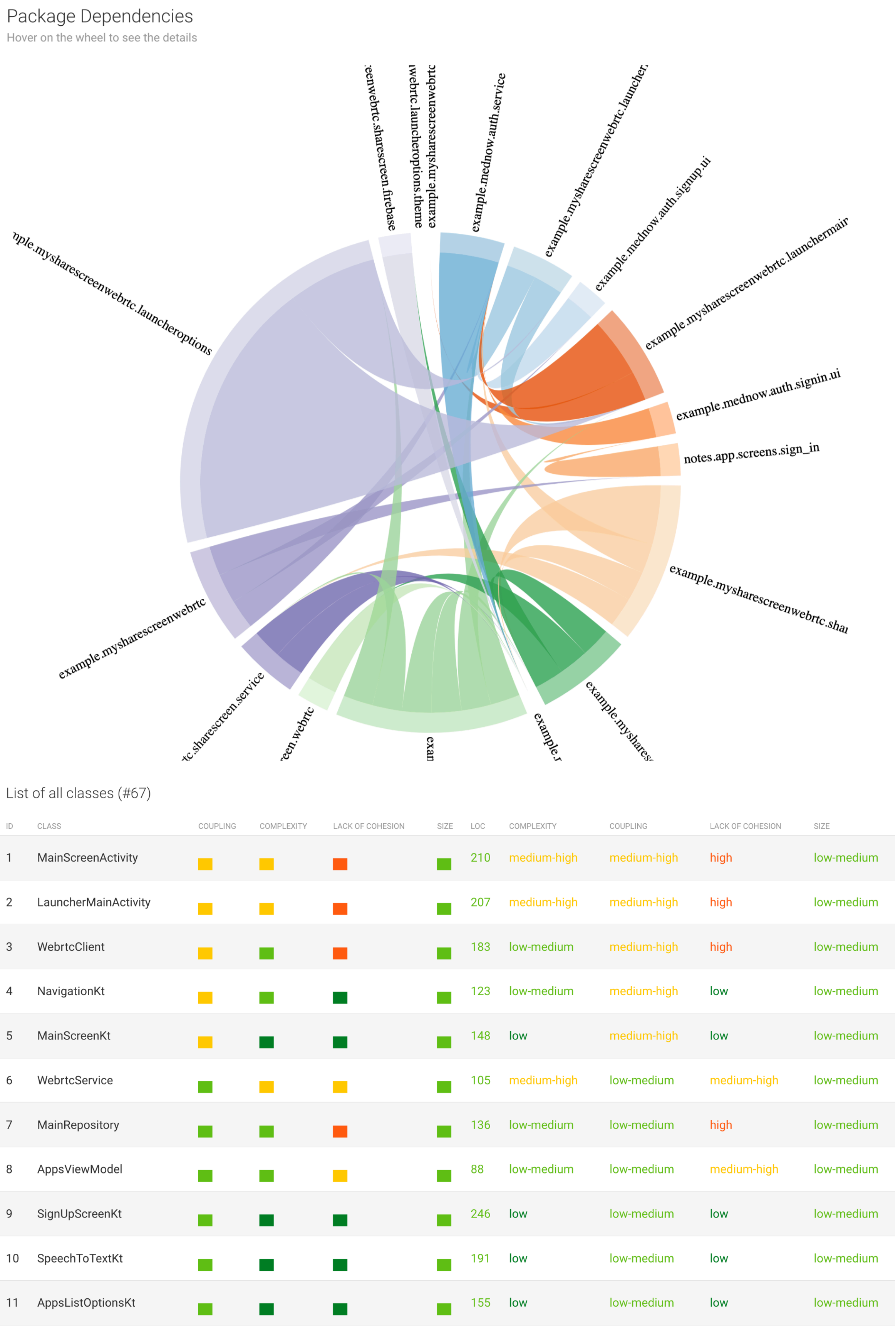
click to zoom, for detailed metrics, press ctrl or ⌘ while hovering

com.notes.app.screens.sign_in
com.example.mysharescreenwebrtc
com.example.mysharescreenwebrtc.widgets
com.example.mysharescreenwebrtc.sharemainscreen
com.example.mednow.auth.signup.ui
com.example.mysharescreenwebrtc.launcheroptions

Lipan Andreea-Alexandra

UBB FMI Software Engineering Master

20 March 2025



Package Dependencies

Hover on the wheel to see the details

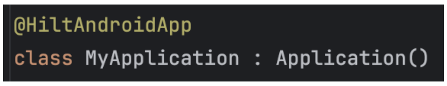
ːeenwebrtc.sharescreen.firebase
ıwebrtc.launcheroptions.theme
example.mysharescreenwebrtc
example.mednow.auth.service
example.mysharescreenwebrtc.launcher.
example.mednow.auth.signup.ui
example.mysharescreenwebrtc.launchermair
example.mednow.auth.signin.ui
notes.app.screens.sign_in
example.mysharescreenwebrtc.shaı
example.myshares
example.ı
exan
een.webrtc
ːc.sharescreen.service
example.mysharescreenwebrtc
ıple.mysharescreenwebrtc.launcheroptions

List of all classes (#67)

ID	CLASS	COUPLING	COMPLEXITY	LACK OF COHESION	SIZE	LOC	COMPLEXITY	COUPLING	LACK OF COHESION	SIZE
1	MainScreenActivity	🟨	🟨	🟧	🟩	210	medium-high	medium-high	high	low-medium
2	LauncherMainActivity	🟨	🟨	🟧	🟩	207	medium-high	medium-high	high	low-medium
3	WebrtcClient	🟨	🟩	🟧	🟩	183	low-medium	medium-high	high	low-medium
4	NavigationKt	🟨	🟩	🟩	🟩	123	low-medium	medium-high	low	low-medium
5	MainScreenKt	🟨	🟩	🟩	🟩	148	low	medium-high	low	low-medium
6	WebrtcService	🟩	🟨	🟨	🟩	105	medium-high	low-medium	medium-high	low-medium
7	MainRepository	🟩	🟩	🟧	🟩	136	low-medium	low-medium	high	low-medium
8	AppsViewModel	🟩	🟩	🟨	🟩	88	low-medium	low-medium	medium-high	low-medium
9	SignUpScreenKt	🟩	🟩	🟩	🟩	246	low	low-medium	low	low-medium
10	SpeechToTextKt	🟩	🟩	🟩	🟩	191	low	low-medium	low	low-medium
11	AppsListOptionsKt	🟩	🟩	🟩	🟩	155	low	low-medium	low	low-medium

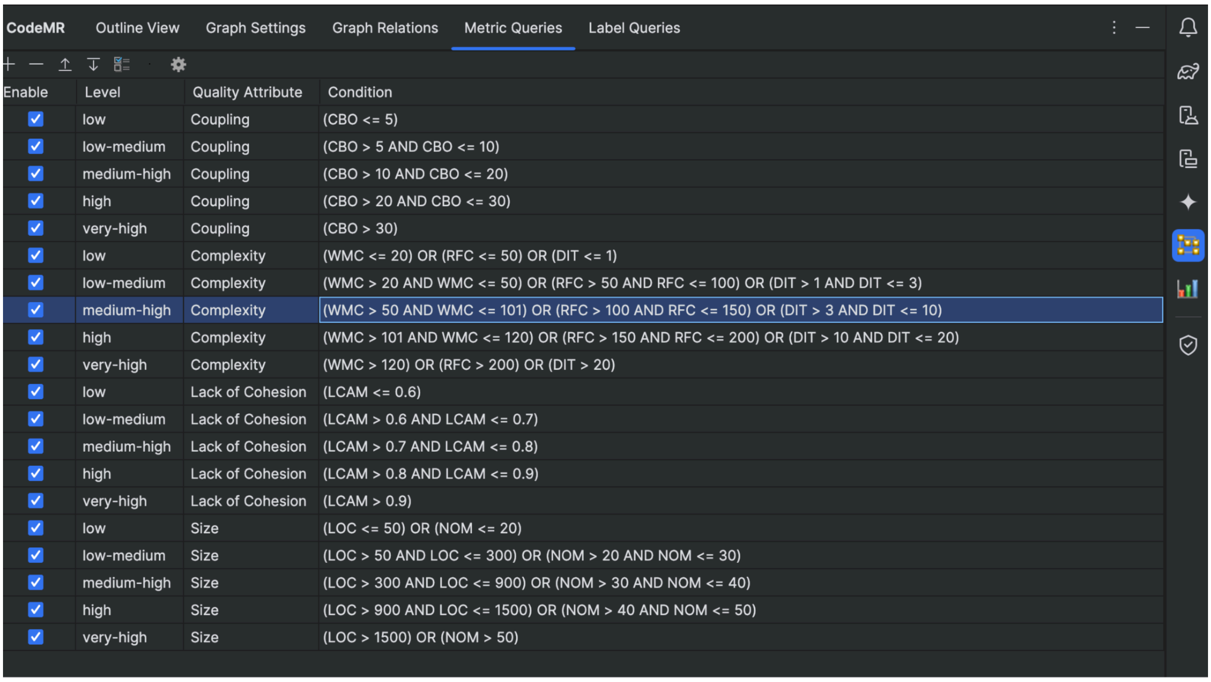
I think the easiest way to actually see the issues is in this table above, seeing exactly what class has what problems. According to these metrics most of my classes were okay with only a few very problematic from a cohesion and coupling standpoint, which I 100% agree with, those classes do need some refactoring.

But again i can not click on any of those lines, can not get any extra information as to why a class was classified a certain way. I have somewhere a class that has 2 lines of code and it's classified as medium-high. why?? The class in question is below.



```
@HiltAndroidApp
class MyApplication : Application()
```

I have found some options in the IDE for this tool (bear in mind i found these by luck because the tool has NO tutorials on the website)

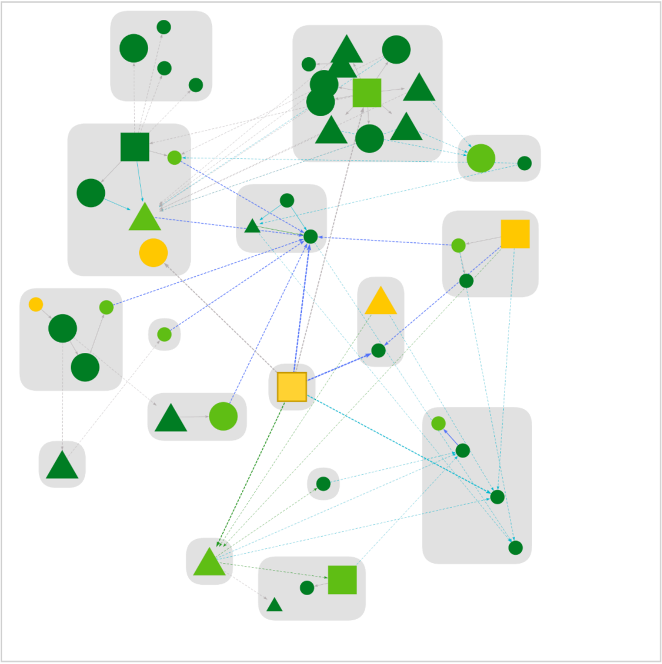


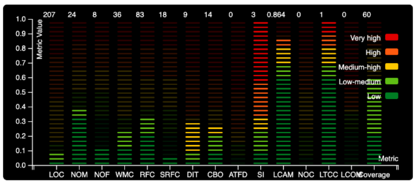
Enable	Level	Quality Attribute	Condition
☑	low	Coupling	(CBO <= 5)
☑	low-medium	Coupling	(CBO > 5 AND CBO <= 10)
☑	medium-high	Coupling	(CBO > 10 AND CBO <= 20)
☑	high	Coupling	(CBO > 20 AND CBO <= 30)
☑	very-high	Coupling	(CBO > 30)
☑	low	Complexity	(WMC <= 20) OR (RFC <= 50) OR (DIT <= 1)
☑	low-medium	Complexity	(WMC > 20 AND WMC <= 50) OR (RFC > 50 AND RFC <= 100) OR (DIT > 1 AND DIT <= 3)
☑	medium-high	Complexity	(WMC > 50 AND WMC <= 101) OR (RFC > 100 AND RFC <= 150) OR (DIT > 3 AND DIT <= 10)
☑	high	Complexity	(WMC > 101 AND WMC <= 120) OR (RFC > 150 AND RFC <= 200) OR (DIT > 10 AND DIT <= 20)
☑	very-high	Complexity	(WMC > 120) OR (RFC > 200) OR (DIT > 20)
☑	low	Lack of Cohesion	(LCAM <= 0.6)
☑	low-medium	Lack of Cohesion	(LCAM > 0.6 AND LCAM <= 0.7)
☑	medium-high	Lack of Cohesion	(LCAM > 0.7 AND LCAM <= 0.8)
☑	high	Lack of Cohesion	(LCAM > 0.8 AND LCAM <= 0.9)
☑	very-high	Lack of Cohesion	(LCAM > 0.9)
☑	low	Size	(LOC <= 50) OR (NOM <= 20)
☑	low-medium	Size	(LOC > 50 AND LOC <= 300) OR (NOM > 20 AND NOM <= 30)
☑	medium-high	Size	(LOC > 300 AND LOC <= 900) OR (NOM > 30 AND NOM <= 40)
☑	high	Size	(LOC > 900 AND LOC <= 1500) OR (NOM > 40 AND NOM <= 50)
☑	very-high	Size	(LOC > 1500) OR (NOM > 50)

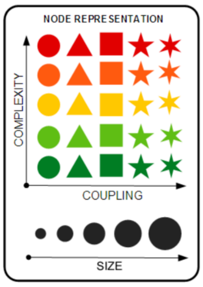
I appreciate that you can change these metrics, that's useful. And I found out why my class with 2 lines has medium-high complexity. Apparently it's because of the Depth of The Inheritance Tree which is greater than 3 (but that's because we inherit Application, which I NEED TO DO). So basically this is a false positive.

The software generates some graphs as well. This one is the most useful imo. It shows the dependencies between packages. It kinda shocked me, seeing how many packages that shouldn't be connected, but are actually connected. This code definitely needs a redesign.

Now I don't know if I'm missing something but some files that were in the same package do not show up on the diagram as being from the same package which is definitely confusing and makes the graph harder to understand. Clicking on a class displays, on the right side, a table with the values for each metric for that class. Same values depicted earlier in the table.







Advantages of using this tool

These metrics are really useful and diagrams are nice ways of displaying it. I like the graphs the most. I think it makes it a clear way to see how interconnected components are. I think also being able to tweak the metrics is useful, but you gotta know what you are doing for that. I haven't used all the metrics, actually i only really cared about 3 or 4 for my project.u8itrgy

I must say it was very easy to install, compared to other softwares.

Disadvantages of using this tool

So many small functionalities that are missing. Why are there any tutorials on the official page? Why can't I hover to see the definition of a metric? Why don't the metrics have more detailed explanations and a clear reason for why it exists? Why does the app offer 0 explanation for why a certain class has a certain value for a certain metric? (I understand you can see some of these in the IDE but it would be so much easier to see it in the report).

Conclusions

I think in general using tools for different Metrics is good and it definitely helped me find problems in my code (on top of the ones I found in the first assignment), mostly regarding coupling and cohesion. But I did have higher hopes, from having used SonarQube for the first assignment. This specific software has useful diagrams and graphs to display the values of different metrics, I agree, but it lacks so many quality of life features that it really makes me not want to use it. All I have to say is that for a paid service (15 Euros a month!!), do better!

Bibliography

https://plugins.jetbrains.com/plugin/10811-codemr