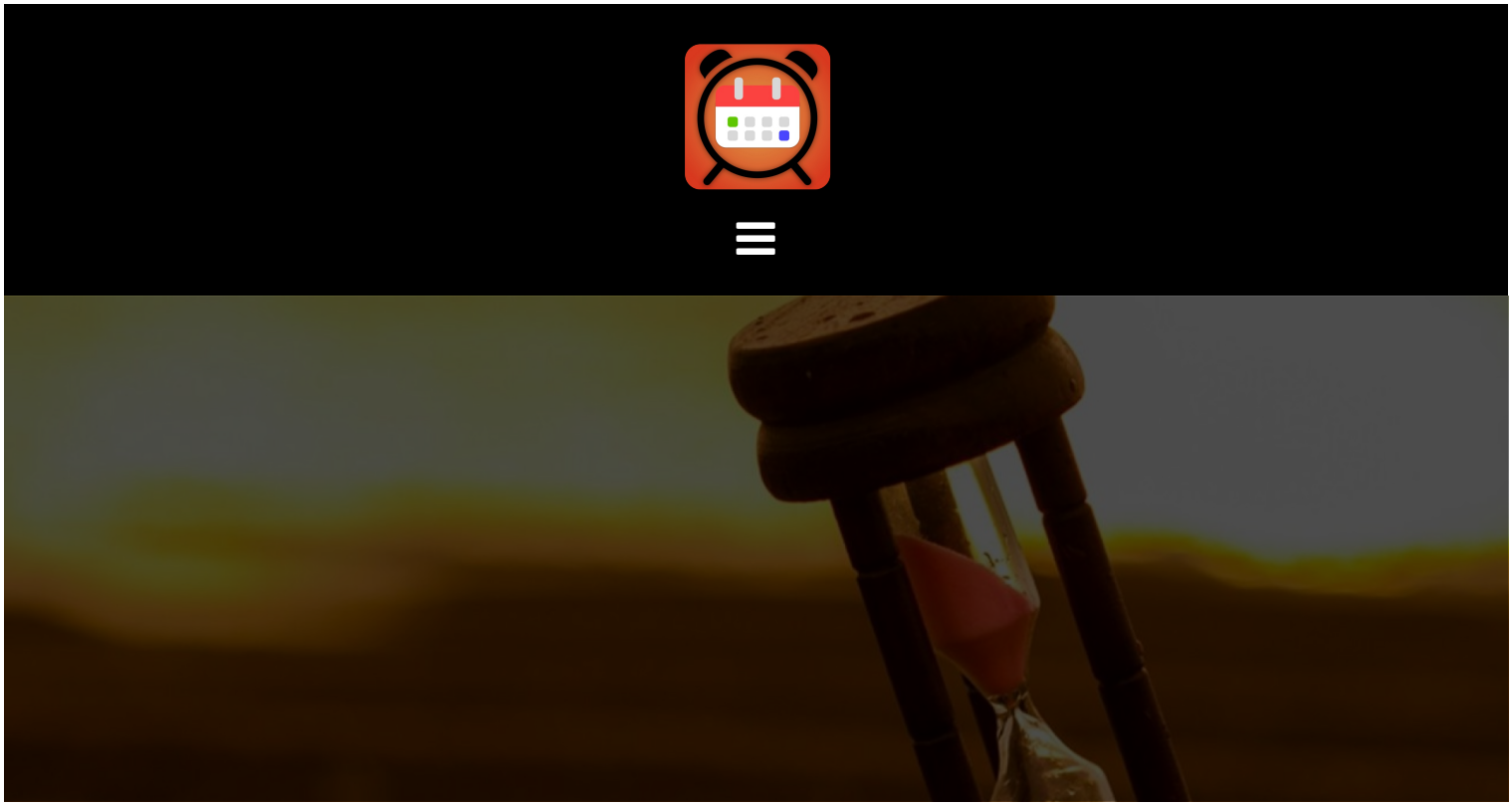


Metrics

Hey guys,

this week we were asked to analyze our code base using a metrics tool.

Our analysis is based on two tools. The first one is the Command Line Tool Taylor working together with Xcode. Due to two the problem that Taylor only produces JSON as the analysis result we decided to develop an own Command Line Tool based on Nodejs around it named *Taylor-Parser* to show the results nicely in the browser.
Check out our tool! (on **github** or **npm**)

We analyzed our code and decided to look at the issues with the following two metrics:

- Too Many Methods (TMM)
  Number of methods in a class must not exceed given limit
- Cyclomatic Complexity (CC)
  Methods must not exceed an admitted value of Cyclomatic Complexity (check for coding example)
  Complexity is determined by the number of decision points in a method plus one for the method entry.

Our self written tool gave out the following report:

This is your result

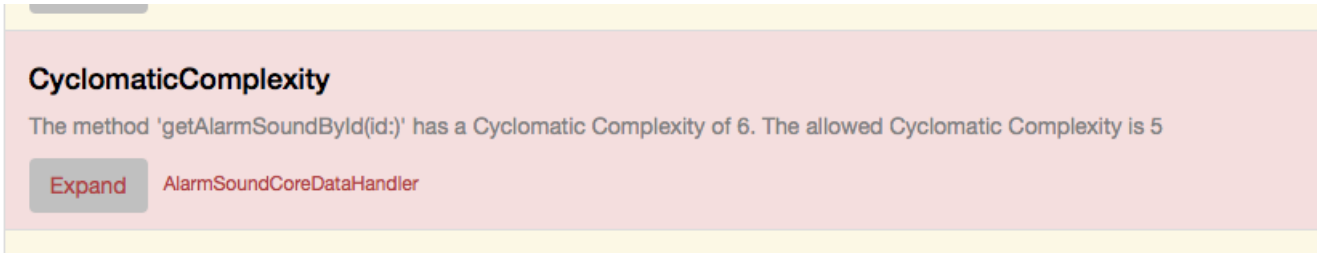
33 Violations!

Rule	Violations
Excessive Class Length	0
Excessive Method Length	7
Too Many Methods	3
Cyclomatic Complexity	19
Nested Block Depth	2
N-Path Complexity	1
Excessive Parameter List	1

We picked the following:

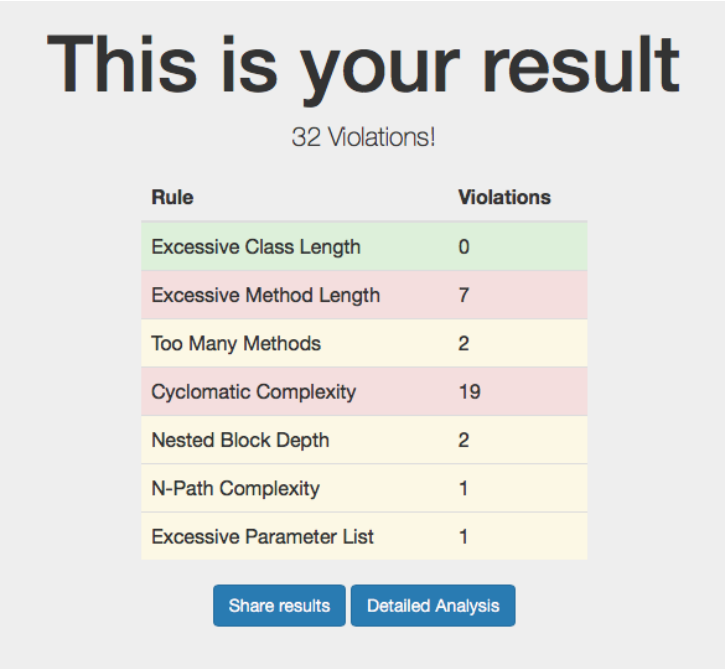
and merged two into one to confirm to the maximum of 10 methods per class.
(See **this** commit with the changes)
This really improved the clarity of the class!

The tool also marked the following:



But we decided not to change it because the Cyclomatic Complexity results from Apple's Core Data Mechanisms which *AlarmSoundCoreDataHandler* wraps.

This is our new results after the changes:



32 left yeay 😀

We don't use automated deployment due to the fact that deployment means publishing to the app store in our case. That means that our tool won't be part of the deployment process!

Warm regards

← Patterns

Test & Installation →

3 thoughts on "Metrics"

Eynorey says:

1. June 2017 at 07:26

Hey there simplistic habitanters,
good to see you putting that much passion and effort into the project! You even wrote your own tool as there wasn't any for your environment.
Since you provided it on Github or via npm also, it'll likely help others in the future who find themselves in the same situation as you. So good work!
Also nice that you decided to check cyclomatic complexity (some cross-course knowledge going on here? 😉
Anyhow, I got why you decided not to do something about it and instead will focus on the other fields to improve your code.
Kepp up the good work!

Greetings from Eynorey@SAM

Reply

Louisa says:

1. June 2017 at 07:40

Hey guys,
it looks pretty good! Cool, that you create your own tool!
You fulfil all of the grading criterias! You handed in code snippets, refactored your code and explained the metrics you use.
May you do well,

Louisa

Reply

Tom bendrath says:

1. June 2017 at 07:42

Hey guys,

your metrics look great! I like the way you used your own command line tool which works fine with NodeJS. Even though, I am wondering if your metrics is part of your deployment? Moreover, you fulfilled the grading criteria by using 2 different metrics.

You linked your changes and also reached some improvement before-after. Additionally, you gave a comprehensive reason why you didn't change your code regarding the cyclomatic complexity violation.
By the way, you guys may want to chill a little bit when coding to prevent those excessive method length violations 😀

Kind regards,
Tom

Reply

Leave a Reply

Your email address will not be published. Required fields are marked *

Name *

Email *

Website

Comment

POST COMMENT