# Assignment 2 - Report

## *CODE REFACTORING*

After a 4-week development period, we observed, using the MetricsTree plugin available in IntelliJ's marketplace, that various project metrics were outside of the ideal range. The ones that we deemed as most important were the *Number of Methods* and *Maintainability Index, Response for a class*, *Weighted methods per class* & *Cyclomatic complexity*.

The *Number of Methods* was alerted by the plugin in all of the more complex classes. We have refactored as much as we could in order to incorporate a maximum of 10 methods inside a class. Here are the methods and classes along with the changes that have been done to improve code metrics:

### 1. Classes

**UserService**

In the first version of the app, the service responsible for creating a link between the APIs and the database contained 18 methods, which meant that it was hard to find a method you were looking for. The *UserService* was a huge class that was creating a blob class smell, while also being hard to maintain and further test.

UserService has been divided into 2 components: the *UserService* that is still in use and the *UserTimeService*, each of which has a unique set of implementations. We also removed all the APIs that would only use one setter at a time and reformatted the method that updates the user to do all the changes into one single method. This reduced the original class's number of methods and improved readability (depending on the kind of method you want to add or modify, you can refer to the particular service). In consequence, by doing this the overall *Maintainability Index* has been lowered from around 50 to an average of 40 between all the new classes and lowered the number of methods to 10.

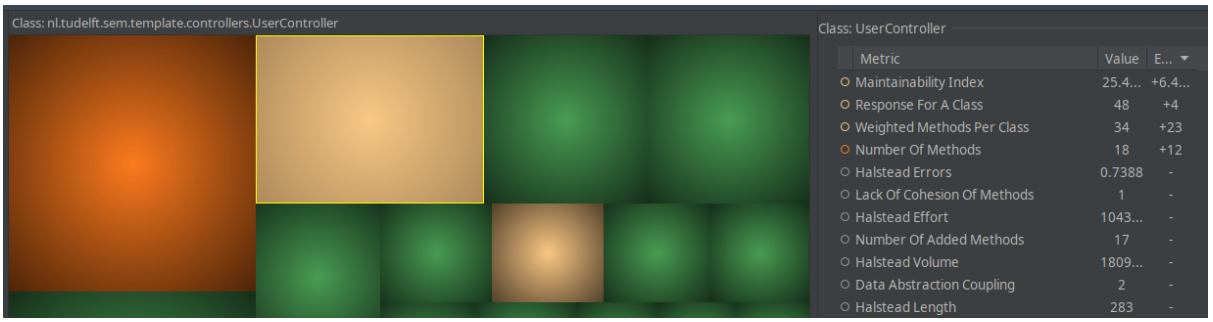
**UserController**

In the EventController two metrics have been improved. Namely the *Number of methods* and *Response for a class* - the number of foreign methods called by the controller.
This has been done by:

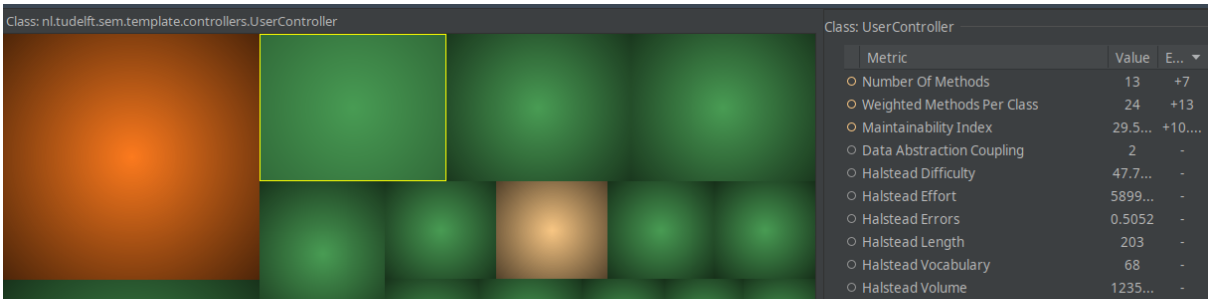
1. deleting endpoints for editing a single field in the user class. Before that we would have for instance separate method for editing the name of the user and a separate method for editing the name of the organisation he/she belonges to.
2. merging those methods. Those methods have been incorporated into one method to update many fields of a user at once. If the client wants to just update one field that is still possible because the method does not need all the arguments in order to function.

The main code smell that was eliminated was the “blob class” and “duplicate code”. The functionalities of those methods have been inserted into one method removing the need to maintain may methods in the future. The following screenshots show the metrics that have been improved:

**BEFORE**



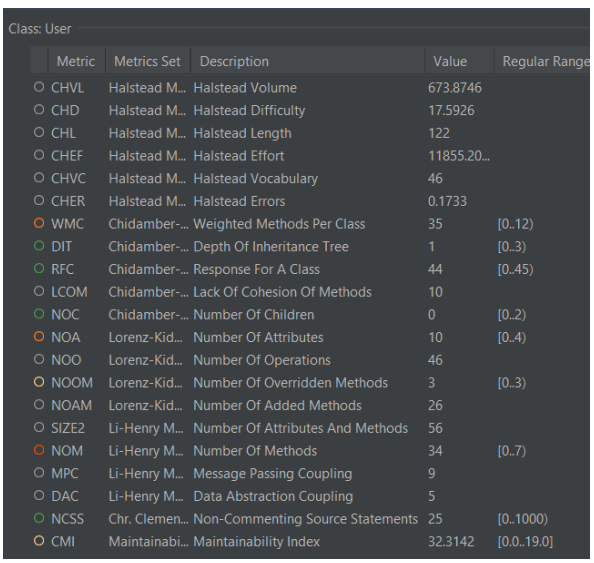

**AFTER**

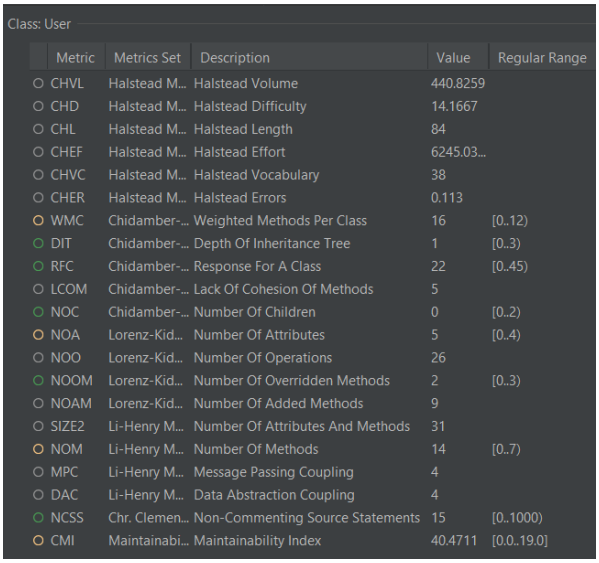



**User**

In the User class we focused on improving the number of methods and number of attributes, since those metrics were the most out of range. The biggest problem was the large number of attributes for which we had getters and setters, which as a consequence increased the number of methods as well. The main changes to improve these metrics were:

1. creating the UserInfo class to contain the user's personal information, reducing the number of attributes of User significantly
2. creating a converter for the UserInfo class to be able to store it in the database
3. removing a constructor, methods relating to changing the schedule and methods that were never used




**BEFORE** **AFTER**

**EventService**

In the event service we focused on improving weighted methods for class and response for a class as those metrics were out of their regular range. We noticed that we used a lot of getters and setters along with some complex if statements to handle eligibility of the user to enroll for the event. Main changes that helped with minimizing the metrics were:

1.) changing long if statements list in updateById method into one method called merge that was added to Event class that handles updating all relevant fields of the event and creates a copy of it

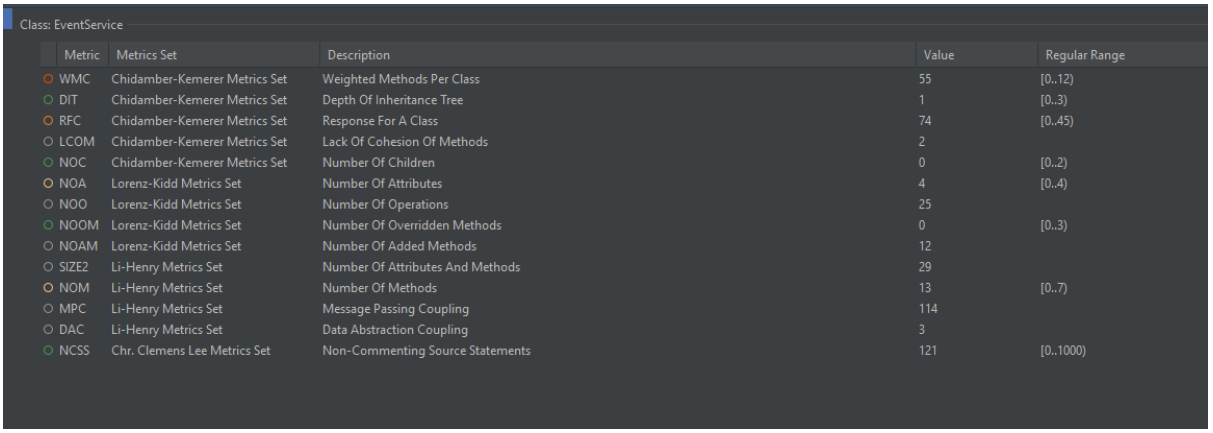
2.) creating utility class ValidityChecker that handles all the logic connected with checking if the user can join the event as well as if the user info is filled correctly and can be used to match the user to events.

3.) creating utility class PositionMatcher that handles filtering all events to end up only with all the events that can be joined by a user.

Overall our changes enabled us to reduce the following code smells:

1. Overcomplicated class by reducing the number of weighted methods per class
2. Duplicate code by extracting all method into util classes that can be reusable

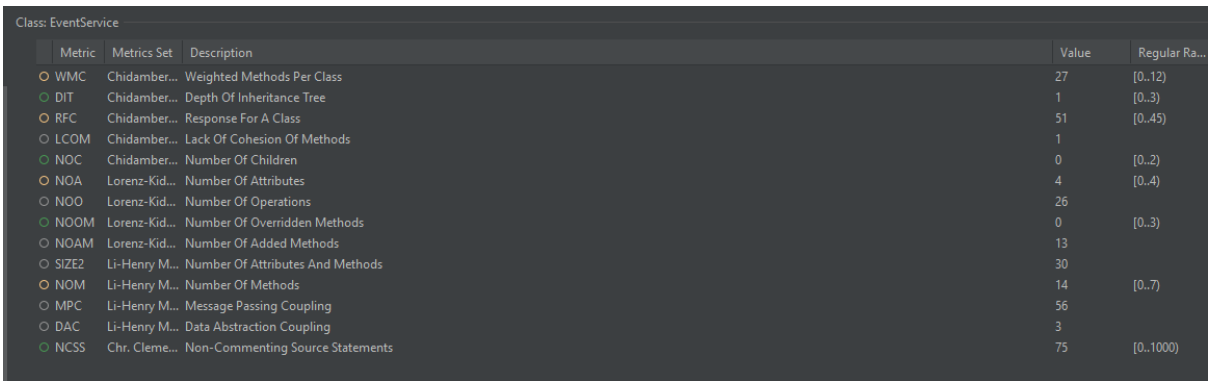
**BEFORE**



Class: EventService

| Metric | Metrics Set | Description | Value | Regular Range |
|---|---|---|---|---|
| WMC | Chidamber-Kemerer Metrics Set | Weighted Methods Per Class | 55 | [0..12) |
| DIT | Chidamber-Kemerer Metrics Set | Depth Of Inheritance Tree | 1 | [0..3) |
| RFC | Chidamber-Kemerer Metrics Set | Response For A Class | 74 | [0..45) |
| LCOM | Chidamber-Kemerer Metrics Set | Lack Of Cohesion Of Methods | 2 | |
| NOC | Chidamber-Kemerer Metrics Set | Number Of Children | 0 | [0..2) |
| NOA | Lorenz-Kidd Metrics Set | Number Of Attributes | 4 | [0..4) |
| NOO | Lorenz-Kidd Metrics Set | Number Of Operations | 25 | |
| NOOM | Lorenz-Kidd Metrics Set | Number Of Overridden Methods | 0 | [0..3) |
| NOAM | Lorenz-Kidd Metrics Set | Number Of Added Methods | 12 | |
| SIZE2 | Li-Henry Metrics Set | Number Of Attributes And Methods | 29 | |
| NOM | Li-Henry Metrics Set | Number Of Methods | 13 | [0..7) |
| MPC | Li-Henry Metrics Set | Message Passing Coupling | 114 | |
| DAC | Li-Henry Metrics Set | Data Abstraction Coupling | 3 | |
| NCSS | Chr. Clemens Lee Metrics Set | Non-Commenting Source Statements | 121 | [0..1000) |

**AFTER**



Class: EventService

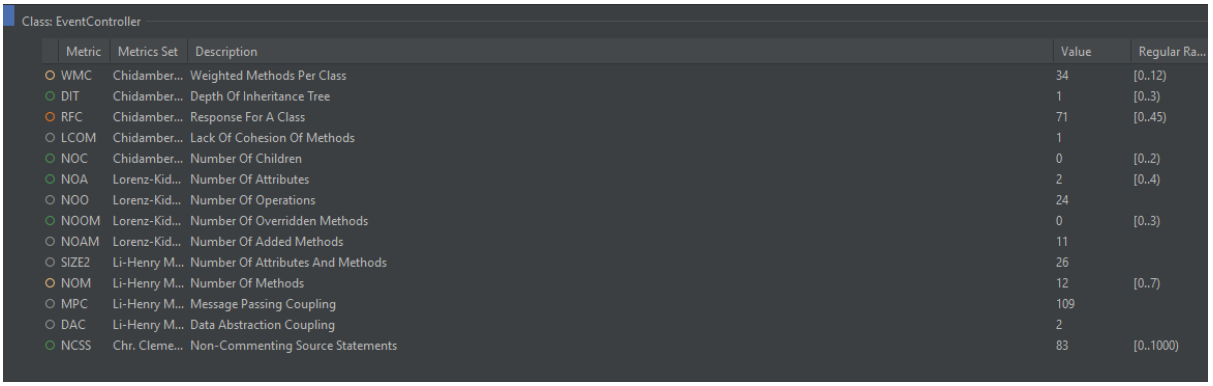
| Metric | Metrics Set | Description | Value | Regular Ra... |
|---|---|---|---|---|
| WMC | Chidamber... | Weighted Methods Per Class | 27 | [0..12) |
| DIT | Chidamber... | Depth Of Inheritance Tree | 1 | [0..3) |
| RFC | Chidamber... | Response For A Class | 51 | [0..45) |
| LCOM | Chidamber... | Lack Of Cohesion Of Methods | 1 | |
| NOC | Chidamber... | Number Of Children | 0 | [0..2) |
| NOA | Lorenz-Kid... | Number Of Attributes | 4 | [0..4) |
| NOO | Lorenz-Kid... | Number Of Operations | 26 | |
| NOOM | Lorenz-Kid... | Number Of Overridden Methods | 0 | [0..3) |
| NOAM | Lorenz-Kid... | Number Of Added Methods | 13 | |
| SIZE2 | Li-Henry M... | Number Of Attributes And Methods | 30 | |
| NOM | Li-Henry M... | Number Of Methods | 14 | [0..7) |
| MPC | Li-Henry M... | Message Passing Coupling | 56 | |
| DAC | Li-Henry M... | Data Abstraction Coupling | 3 | |
| NCSS | Chr. Cleme... | Non-Commenting Source Statements | 75 | [0..1000) |

**EventController**

In the case of EventController we also noticed too high a value of *Weighted methods per class* and *Response for a class*, along with the number of methods. The main reason for such a high score was the number of operations handled in a controller that should have been handled by services. We also noticed that some of the methods do identical things but for different parameters so we decided to merge getEventsByUser, matchEvents and getEvents into one method that handled different options according to RequestParams such as "owner" and "match". Also we saw that the getRequests method is redundant as the user can just check the queue of the event by requesting the event by id. Similarly accept and reject methods were merged into one that takes additional RequestParam outcome to indicate whether the user is to be rejected or accepted. This helped with reducing duplicated code as both of the methods were identical in terms of exception handling and notification sending.

Also we changed the previously used WebClient to restTemplate to communicate with different microservices and moved all communication to the EventService class. This helped us with making our methods easier to maintain as we don't require any logic that handle setting up the WebClient, because all of that is done in the service using RestTemplate config that we use throughout our application. In the end we managed to reduce the value of all the metrics significantly, and the new value of *Response for a class* metric is in its regular range. By implementing all above mentioned changes we also reduced the OverComplicated Classes code smell by extracting all.

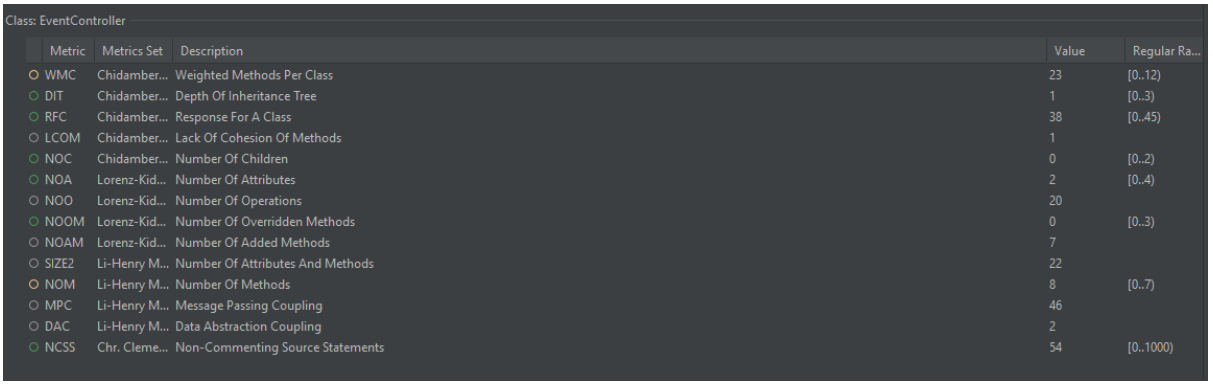
**BEFORE**



Class: EventController

| Metric | Metrics Set | Description | Value | Regular Ra... |
|---|---|---|---|---|
| WMC | Chidamber... | Weighted Methods Per Class | 34 | [0..12) |
| DIT | Chidamber... | Depth Of Inheritance Tree | 1 | [0..3) |
| RFC | Chidamber... | Response For A Class | 71 | [0..45) |
| LCOM | Chidamber... | Lack Of Cohesion Of Methods | 1 | |
| NOC | Chidamber... | Number Of Children | 0 | [0..2) |
| NOA | Lorenz-Kid... | Number Of Attributes | 2 | [0..4) |
| NOO | Lorenz-Kid... | Number Of Operations | 24 | |
| NOOM | Lorenz-Kid... | Number Of Overridden Methods | 0 | [0..3) |
| NOAM | Lorenz-Kid... | Number Of Added Methods | 11 | |
| SIZE2 | Li-Henry M... | Number Of Attributes And Methods | 26 | |
| NOM | Li-Henry M... | Number Of Methods | 12 | [0..7) |
| MPC | Li-Henry M... | Message Passing Coupling | 109 | |
| DAC | Li-Henry M... | Data Abstraction Coupling | 2 | |
| NCSS | Chr. Cleme... | Non-Commenting Source Statements | 83 | [0..1000) |

**AFTER**



Class: EventController

| Metric | Metrics Set | Description | Value | Regular Ra... |
|---|---|---|---|---|
| WMC | Chidamber... | Weighted Methods Per Class | 23 | [0..12) |
| DIT | Chidamber... | Depth Of Inheritance Tree | 1 | [0..3) |
| RFC | Chidamber... | Response For A Class | 38 | [0..45) |
| LCOM | Chidamber... | Lack Of Cohesion Of Methods | 1 | |
| NOC | Chidamber... | Number Of Children | 0 | [0..2) |
| NOA | Lorenz-Kid... | Number Of Attributes | 2 | [0..4) |
| NOO | Lorenz-Kid... | Number Of Operations | 20 | |
| NOOM | Lorenz-Kid... | Number Of Overridden Methods | 0 | [0..3) |
| NOAM | Lorenz-Kid... | Number Of Added Methods | 7 | |
| SIZE2 | Li-Henry M... | Number Of Attributes And Methods | 22 | |
| NOM | Li-Henry M... | Number Of Methods | 8 | [0..7) |
| MPC | Li-Henry M... | Message Passing Coupling | 46 | |
| DAC | Li-Henry M... | Data Abstraction Coupling | 2 | |
| NCSS | Chr. Cleme... | Non-Commenting Source Statements | 54 | [0..1000) |

## 2. Methods

### UpdateById - UserService

```
public Optional<User> updateById(Long id, String name, String organization,
String gender, Certificate certificate, List<Position> positions) {
```

Here we have focused on the method update a user by their Id in the UserService of the User microservice which was responsible for telling the JPA repository to update the fields of a particular user. This method had two code smells.

1. long parameter list - previously the method would have all the parameters that is the: name, organization, gender, certificate, and list of positions. To eliminate this an additional object has been created to store those parameters namely the **UserModel**. This way the method has only two parameters: id and the usermodel.
2. long method - The user that was supposed to be updated was retrieved many times from the Optional object, this introduced more complexity and also made the code longer so instead of this we saved the optional do a User object and then performed the necessary changes. It reduced the lines of code from 42 to 31.

```
Optional<User> toUpdate = getById(id);
toUpdate.get();
```

**BEFORE**

Method: updateById(Long, String, String, String, Certificate, List<Position>)

| Metric | Metrics Set | Description | Value | Regular Range |
|---|---|---|---|---|
| CND | | Condition Nesting Depth | 2 | [0..2) |
| LND | | Loop Nesting Depth | 0 | [0..2) |
| CC | | McCabe Cyclomatic Complexity | 7 | [0..3) |
| NOL | | Number Of Loops | 0 | |
| LOC | | Lines Of Code | 42 | [0..11) |
| NOPM | | Number Of Parameters | 6 | [0..3) |
| HVL | Halstead Metric Set | Halstead Volume | 433.8179 | |
| HD | Halstead Metric Set | Halstead Difficulty | 21.6471 | |
| HL | Halstead Metric Set | Halstead Length | 86 | |
| HEF | Halstead Metric Set | Halstead Effort | 9390.8815 | |
| HVC | Halstead Metric Set | Halstead Vocabulary | 33 | |
| HER | Halstead Metric Set | Halstead Errors | 0.1484 | |
| MMI | Maintainability Index | Maintainability Index | 45.868 | [0.0..19.0] |

**AFTER**

Method: updateById(Long, UserModel)

| Metric | Metrics Set | Description | Value | Regular Range |
|---|---|---|---|---|
| CND | | Condition Nesting Depth | 2 | [0..2) |
| LND | | Loop Nesting Depth | 0 | [0..2) |
| CC | | McCabe Cyclomatic Complexity | 7 | [0..3) |
| NOL | | Number Of Loops | 0 | |
| LOC | | Lines Of Code | 31 | [0..11) |
| NOPM | | Number Of Parameters | 2 | [0..3) |
| HVL | Halstead Metric Set | Halstead Volume | 281.7629 | |
| HD | Halstead Metric Set | Halstead Difficulty | 25.3636 | |
| HL | Halstead Metric Set | Halstead Length | 58 | |
| HEF | Halstead Metric Set | Halstead Effort | 7146.5317 | |
| HVC | Halstead Metric Set | Halstead Vocabulary | 29 | |
| HER | Halstead Metric Set | Halstead Errors | 0.1237 | |
| MMI | Maintainability Index | Maintainability Index | 50.0599 | [0.0..19.0] |

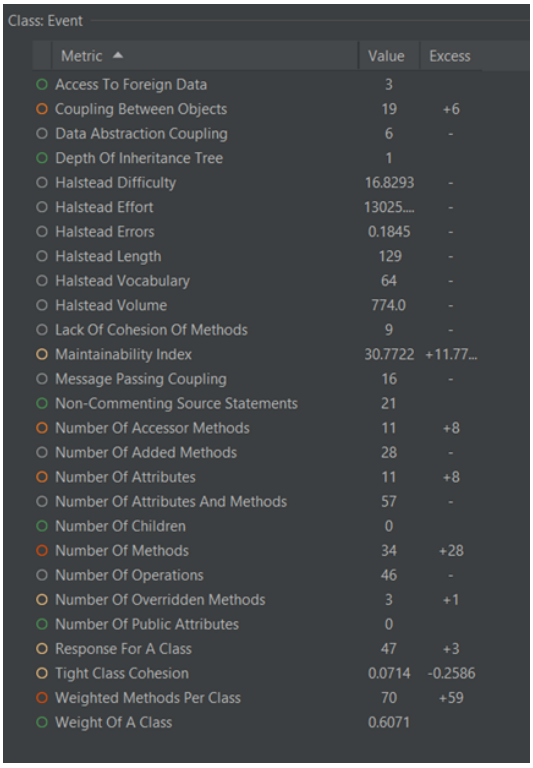
**Event - equals and hashcode**

The Event class has a large number of attributes compared to other classes. This caused the *equals* and *hashcode* methods to have a very high *Cyclomatic Complexity*, raising the *Weighted Methods per Class* metric to 70. Since events have unique and non-null labels, we were able to replace these methods generated by lombok with custom versions that make use of the label uniqueness:

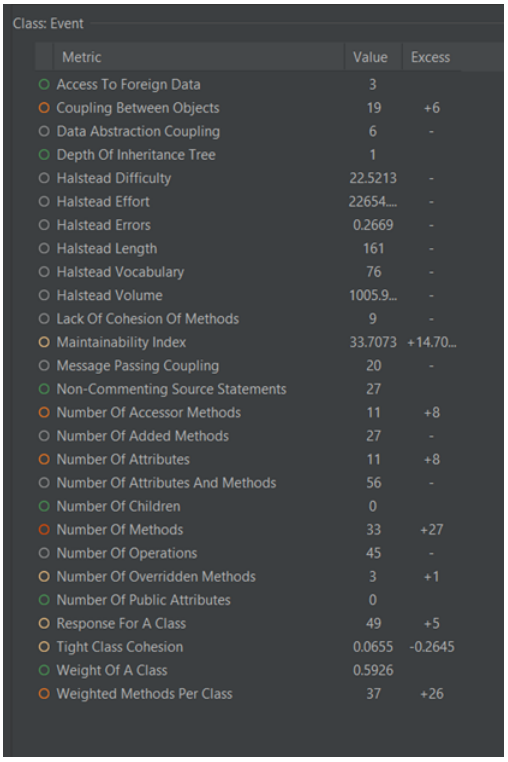
1. The *equals* method used to compare every attribute of the two events before returning true. It now only checks the label.
2. The *hashCode* method also used every attribute before, whereas now it only hashes the label.

Even though the Event object has a unique numerical identifier, we chose to use the label for these specific methods because there is a constructor for Event that doesn't require an id. These are used in some of our tests to check if event creation works. The label, on the other hand, is present for every event. Changing these methods alone decreased the *Weighted Methods per Class* metric by 33.

The *Cyclomatic Complexity* of the equals method was reduced from 25 to 4, and from the hashCode method from 12 to 1.




**BEFORE**




**AFTER**

Finally, we made a few minor changes to the general code, such as removing commented lines of code, correcting grammatical errors in the documentation, and getting rid of console printouts that were used to debug the course of any procedure.

**Enqueue - Event (shared)**

The enqueue method used to check if the position a user wants to fill is actually available. However, this is something we were already checking in the EventService. Therefore, we were able to remove redundant lines of code and prevent the system from doing unnecessary extra work.

**BEFORE**

```
public boolean enqueue(String name, PositionName position) {
    if (!positions.contains(position)) {
        return false;
    }

    queue.add(new Request(name, position));
    return true;
}
```

**AFTER**

```
public boolean enqueue(String name, PositionName position) {
    return queue.add(new Request(name, position));
}
```

The *Lines Of Code* were reduced from 14 to 2.

**BEFORE**

Method: enqueue(String, PositionName)

| Metric | Metrics Set | Description | Value | Regular Ra... |
|---|---|---|---|---|
| CND | | Condition Nesting Depth | 1 | [0..2) |
| LND | | Loop Nesting Depth | 0 | [0..2) |
| CC | | McCabe Cyclomatic Complexity | 2 | [0..3) |
| NOL | | Number Of Loops | 0 | |
| LOC | | Lines Of Code | 14 | [0..11) |
| NOPM | | Number Of Parameters | 2 | [0..3) |
| HVL | Halstead M... | Halstead Volume | 60.9177 | |
| HD | Halstead M... | Halstead Difficulty | 4.6667 | |
| HL | Halstead M... | Halstead Length | 16 | |
| HEF | Halstead M... | Halstead Effort | 284.2825 | |
| HVC | Halstead M... | Halstead Vocabulary | 14 | |
| HER | Halstead M... | Halstead Errors | 0.0144 | |
| MMI | Maintainabi... | Maintainability Index | 62.4545 | [0.0..19.0] |

**AFTER**

Method: enqueue(String, PositionName)

| Metric | Metrics Set | Description | Value | Regular R |
|---|---|---|---|---|
| CND | | Condition Nesting Depth | 0 | [0..2) |
| LND | | Loop Nesting Depth | 0 | [0..2) |
| CC | | McCabe Cyclomatic Complexity | 1 | [0..3) |
| NOL | | Number Of Loops | 0 | |
| LOC | | Lines Of Code | 10 | [0..11) |
| NOPM | | Number Of Parameters | 2 | [0..3) |
| HVL | Halstead M... | Halstead Volume | 24.0 | |
| HD | Halstead M... | Halstead Difficulty | 2.5 | |
| HL | Halstead M... | Halstead Length | 8 | |
| HEF | Halstead M... | Halstead Effort | 60.0 | |
| HVC | Halstead M... | Halstead Vocabulary | 8 | |
| HER | Halstead M... | Halstead Errors | 0.0051 | |
| MMI | Maintainabi... | Maintainability Index | 68.5218 | [0.0..19.0] |

**Send notification - NotificationController**

The method that automatically sends notifications to a specific user based on the enqueue request outcome was 32 lines long and had some hard coded microservice ports. This method made use of WebClient to send requests to other microservices, while the rest of the application uses restTemplate. To make the NotificationController cleaner and more robust, while lowering the number of lines of code in the method, we now use restTemplate instead of webclient.

**BEFORE**

Method: sendNotification(Long, String, Outcome)

| Metric | Metrics Set | Description | Value | Regular Ra |
|---|---|---|---|---|
| CND | | Condition Nesting Depth | 1 | [0..2) |
| LND | | Loop Nesting Depth | 0 | [0..2) |
| CC | | McCabe Cyclomatic Complexity | 4 | [0..3) |
| NOL | | Number Of Loops | 0 | |
| LOC | | Lines Of Code | 32 | [0..11) |
| NOPM | | Number Of Parameters | 3 | [0..3) |
| HVL | Halstead M... | Halstead Volume | 412.5315 | |
| HD | Halstead M... | Halstead Difficulty | 16.1905 | |
| HL | Halstead M... | Halstead Length | 77 | |
| HEF | Halstead M... | Halstead Effort | 6679.0815 | |
| HVC | Halstead M... | Halstead Vocabulary | 41 | |
| HER | Halstead M... | Halstead Errors | 0.1182 | |
| MMI | Maintainabi... | Maintainability Index | 48.6707 | [0.0..19.0] |

**AFTER**

Method: sendNotification(Long, String, Outcome)

| Metric | Metrics Set | Description | Value | Regular Ra |
|---|---|---|---|---|
| CND | | Condition Nesting Depth | 1 | [0..2) |
| LND | | Loop Nesting Depth | 0 | [0..2) |
| CC | | McCabe Cyclomatic Complexity | 3 | [0..3) |
| NOL | | Number Of Loops | 0 | |
| LOC | | Lines Of Code | 25 | [0..11) |
| NOPM | | Number Of Parameters | 3 | [0..3) |
| HVL | Halstead M... | Halstead Volume | 284.3459 | |
| HD | Halstead M... | Halstead Difficulty | 10.8182 | |
| HL | Halstead M... | Halstead Length | 55 | |
| HEF | Halstead M... | Halstead Effort | 3076.1054 | |
| HVC | Halstead M... | Halstead Vocabulary | 36 | |
| HER | Halstead M... | Halstead Errors | 0.0705 | |
| MMI | Maintainabi... | Maintainability Index | 52.1795 | [0.0..19.0] |

**EnqueueById - Event Service**

In this method we noticed a very high cyclomatic complexity and high line count. We distinguished that the main functionality of this method is to verify if the user is eligible to join the event using a sequence of if statements. We extracted it into utility class to the method canJoin. In the method we got rid of if statements and used with proper names to then check if all the conditions are met. This reduced the cyclomatic complexity of original method to 4 and lines of code to 23 while in the extracted method we now have cyclomatic complexity of 10 and 20 lines of code. Because of the high number of checks we have to make to it is very hard to reduce the cyclomatic complexity ever further. The only way would be to make a few more methods to check every criterion.

**BEFORE**

Method: enqueueById(Long, User, PositionName, long)

| Metric | Metrics Set | Description | Value | Regular Ra... |
|---|---|---|---|---|
| CND | | Condition Nesting Depth | 1 | [0..2) |
| LND | | Loop Nesting Depth | 0 | [0..2) |
| CC | | McCabe Cyclomatic Complexity | 15 | [0..3) |
| NOL | | Number Of Loops | 0 | |
| LOC | | Lines Of Code | 50 | [0..11) |
| NOPM | | Number Of Parameters | 4 | [0..3) |

**AFTER**

Method: enqueueById(Long, Long, PositionName, long)

| Metric | Metrics Set | Description | Value | Regular Ra... |
|---|---|---|---|---|
| CND | | Condition Nesting Depth | 1 | [0..2) |
| LND | | Loop Nesting Depth | 0 | [0..2) |
| CC | | McCabe Cyclomatic Complexity | 4 | [0..3) |
| NOL | | Number Of Loops | 0 | |
| LOC | | Lines Of Code | 23 | [0..11) |
| NOPM | | Number Of Parameters | 4 | [0..3) |

Method: canJoin(PositionName, long)

| Metric | Metrics Set | Description | Value | Regular Ra... |
|---|---|---|---|---|
| CND | | Condition Nesting Depth | 0 | [0..2) |
| LND | | Loop Nesting Depth | 0 | [0..2) |
| CC | | McCabe Cyclomatic Complexity | 10 | [0..3) |
| NOL | | Number Of Loops | 0 | |
| LOC | | Lines Of Code | 20 | [0..11) |
| NOPM | | Number Of Parameters | 2 | [0..3) |

**getMatchedEvents - eventService**

In this method we focused on reducing cyclomatic complexity and lines of code. What we noticed is that there are two loops that do basically the same thing. We extracted this functionality to the utility class and created matchPositions method. This got rid of the two loops needed in the original method which reduced the lines of code in the original method as well as cyclomatic complexity. Because of the nested loops and if statements in the loop that matched the events we could not get the cyclomatic complexity of the matchPosition method lower than 7 but it still is an improvement in comparison to the previous version of the method.

**BEFORE**

Method: getMatchedEvents(User)

| Metric | Metrics Set | Description | Value | Regular Ra... |
|---|---|---|---|---|
| CND | | Condition Nesting Depth | 1 | [0..2) |
| LND | | Loop Nesting Depth | 2 | [0..2) |
| CC | | McCabe Cyclomatic Complexity | 13 | [0..3) |
| NOL | | Number Of Loops | 4 | |
| LOC | | Lines Of Code | 33 | [0..11) |
| NOPM | | Number Of Parameters | 1 | [0..3) |

**AFTER**

Method: getMatchedEvents(Long)

| Metric | Metrics Set | Description | Value | Regular Ra... |
|---|---|---|---|---|
| CND | | Condition Nesting Depth | 1 | [0..2) |
| LND | | Loop Nesting Depth | 0 | [0..2) |
| CC | | McCabe Cyclomatic Complexity | 2 | [0..3) |
| NOL | | Number Of Loops | 0 | |
| LOC | | Lines Of Code | 18 | [0..11) |
| NOPM | | Number Of Parameters | 1 | [0..3) |

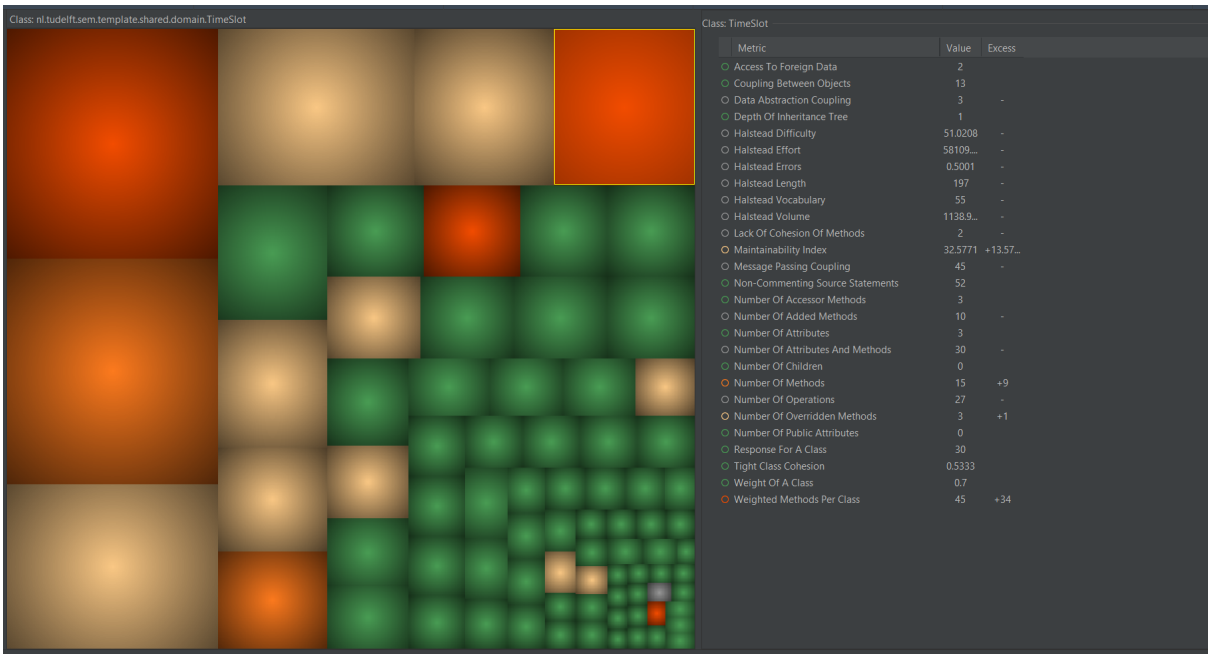
Method: matchPositions(List<Position>, List<Event>, User)

| Metric | Metrics Set | Description | Value | Regular Ra... |
|---|---|---|---|---|
| CND | | Condition Nesting Depth | 1 | [0..2) |
| LND | | Loop Nesting Depth | 2 | [0..2) |
| CC | | McCabe Cyclomatic Complexity | 7 | [0..3) |
| NOL | | Number Of Loops | 2 | |
| LOC | | Lines Of Code | 20 | [0..11) |
| NOPM | | Number Of Parameters | 3 | [0..3) |

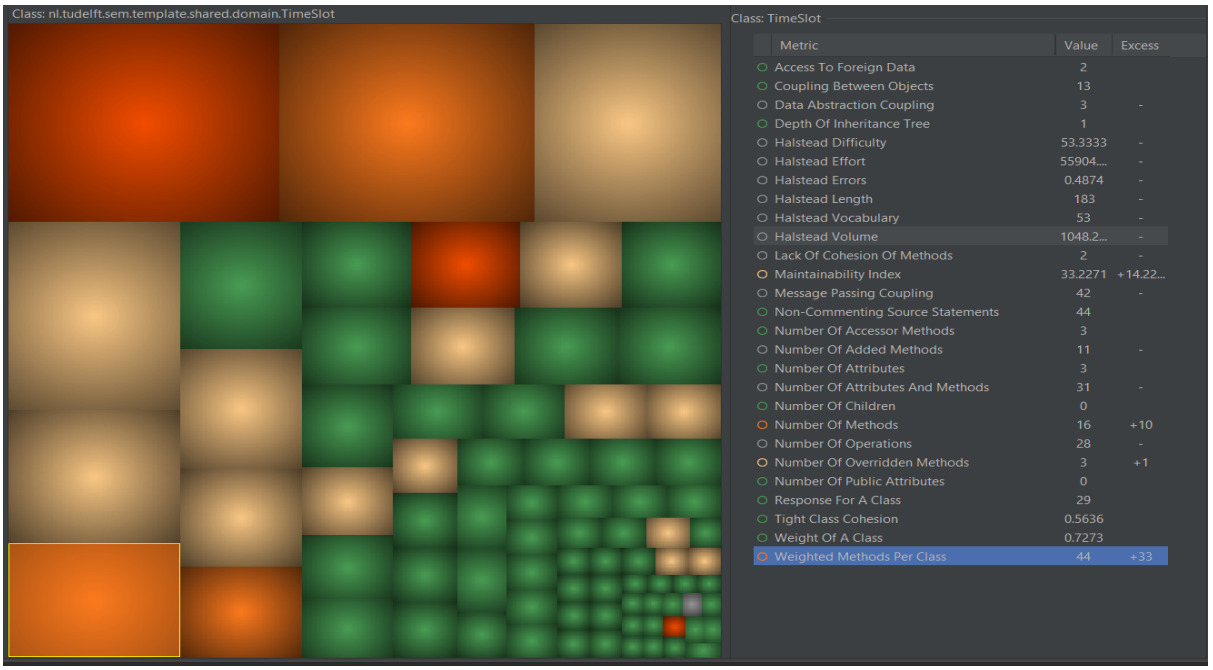
## matchSchedule - TimeSlot

In this method we noticed a very high cyclomatic complexity, duplicated code and an overall long and complicated method(code smell). Initially, because of the structure of the class that is able to also maintain recursive timeslots, we had 2 for loops that had the same functionality. Those were externalized to another function to reduce code duplication and increase maintainability and, thus, reducing the technical dept. Moreover, because we changed how a recurring timeslot is stored (if a slot is recurring, the week is now set to -1, instead of a rand value), it allows us to delete one of the three arrays that was used to handle the slots in the function. This means that we will call the auxiliary function(the one we mentioned we externalized some functionality to it) one time less, making the matchSchedule function easier to understand and maintain.

**BEFORE**



**AFTER**




## *BUG FIXES*

In the previous version, we wouldn't check to see if the time period entered by a rower would actually be practicable before adding a new time slot to a user's schedule. Now, when someone tries to call one of the *addTimeSlot* methods inside the *UserTimeSlotService,* we'll also alert the system when the period's end time is later than its start time or for weeks that have a negative index and disallow incorrect schedules.

Some tests were prepared for checks that did not exist. After also verifying that when adding a new user it doesn’t already exist inside the database, we have created fitting tests for this new feature and also for the one previously mentioned.