Delft University of Technology

Software Engineering Methods

Assignment 2: Refactoring

CSE2115	Group 31b
Alexandra Darie	5511100
Elena Dumitrescu	5527236
Xingyu Han	5343755
Zoya van Meel	5114470
Robert Vădăstreanu	5508096
Sander Vermeulen	5482127

January 17, 2023

Task 1: Software Analytics

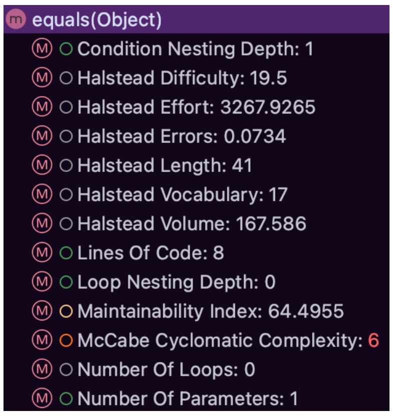
For this assignment, we first had a look over the code, in order to identify methods and classes that presented code smells (mainly, long/complex methods and classes, as well as long parameter lists). To improve our code, we decided to use two metric tools: CodeMR and MetricsTree, since we thought they complement each other very well and provide valuable insights into the state of our code. We used CodeMR for documenting class-level metrics, focusing mainly on cohesion and coupling. The MetricsTree was very helpful in analyzing method-level metrics, such as lines of code and cyclomatic complexity, as well as some more detailed class-level metrics.

1.1 Methods

As stated above, for investigating the methods that presented code smells, we have solely based on the MetricsTree tool.

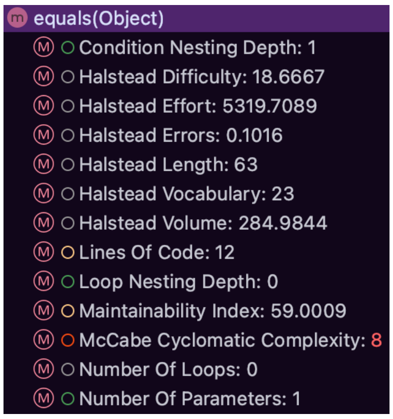
1.1.1 User microservice

- **EventIdModel**: equals(Object) - We have identified that this particular method has a high cyclomatic complexity of 6, whereas the regular range is [0,3).



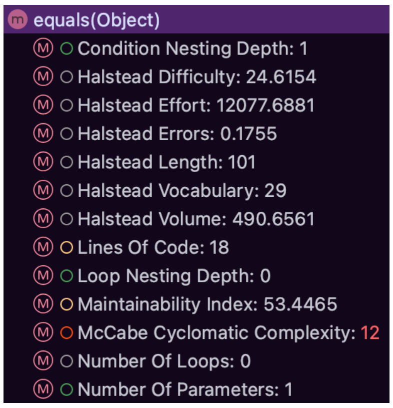
```
equals(Object)
  Condition Nesting Depth: 1
  Halstead Difficulty: 19.5
  Halstead Effort: 3267.9265
  Halstead Errors: 0.0734
  Halstead Length: 41
  Halstead Vocabulary: 17
  Halstead Volume: 167.586
  Lines Of Code: 8
  Loop Nesting Depth: 0
  Maintainability Index: 64.4955
  McCabe Cyclomatic Complexity: 6
  Number Of Loops: 0
  Number Of Parameters: 1
```

- **CustomPair**: equals(Object) - We have identified that this particular method has a high cyclomatic complexity of 8, whereas the regular range is [0,3).



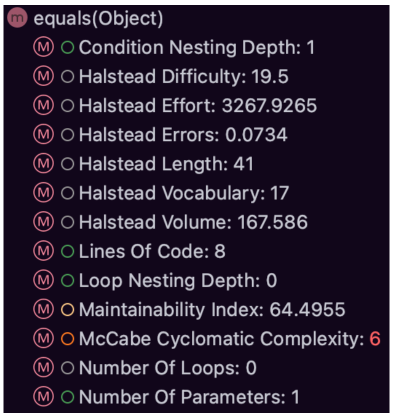
```
equals(Object)
  Condition Nesting Depth: 1
  Halstead Difficulty: 18.6667
  Halstead Effort: 5319.7089
  Halstead Errors: 0.1016
  Halstead Length: 63
  Halstead Vocabulary: 23
  Halstead Volume: 284.9844
  Lines Of Code: 12
  Loop Nesting Depth: 0
  Maintainability Index: 59.0009
  McCabe Cyclomatic Complexity: 8
  Number Of Loops: 0
  Number Of Parameters: 1
```

- **EventModel**: equals(Object) - We have identified that this particular method has a high cyclomatic complexity of 12, whereas the regular range is [0,3).



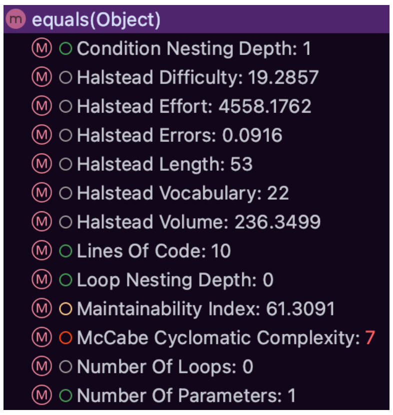
```
equals(Object)
Condition Nesting Depth: 1
Halstead Difficulty: 24.6154
Halstead Effort: 12077.6881
Halstead Errors: 0.1755
Halstead Length: 101
Halstead Vocabulary: 29
Halstead Volume: 490.6561
Lines Of Code: 18
Loop Nesting Depth: 0
Maintainability Index: 53.4465
McCabe Cyclomatic Complexity: 12
Number Of Loops: 0
Number Of Parameters: 1
```

- **UserApprovalModel**: equals(Object) - We have identified that this particular method has a high cyclomatic complexity of 6, whereas the regular range is [0,3).



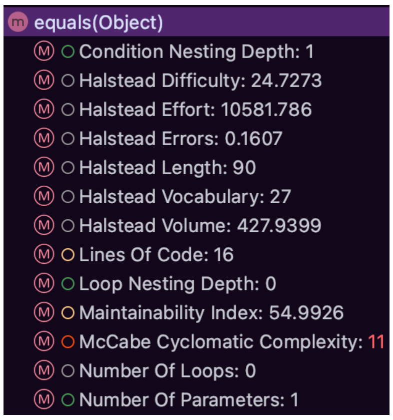
```
equals(Object)
Condition Nesting Depth: 1
Halstead Difficulty: 19.5
Halstead Effort: 3267.9265
Halstead Errors: 0.0734
Halstead Length: 41
Halstead Vocabulary: 17
Halstead Volume: 167.586
Lines Of Code: 8
Loop Nesting Depth: 0
Maintainability Index: 64.4955
McCabe Cyclomatic Complexity: 6
Number Of Loops: 0
Number Of Parameters: 1
```

- **UserInformationModel**: equals(Object) - We have identified that this particular method has a high cyclomatic complexity of 7, whereas the regular range is [0,3).



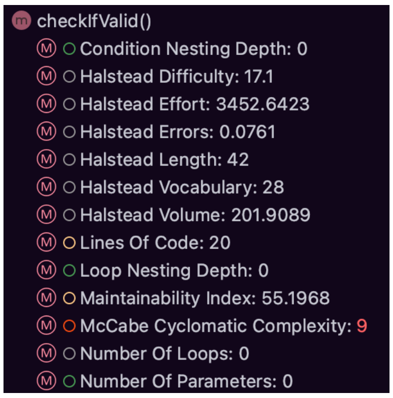
```
equals(Object)
Condition Nesting Depth: 1
Halstead Difficulty: 19.2857
Halstead Effort: 4558.1762
Halstead Errors: 0.0916
Halstead Length: 53
Halstead Vocabulary: 22
Halstead Volume: 236.3499
Lines Of Code: 10
Loop Nesting Depth: 0
Maintainability Index: 61.3091
McCabe Cyclomatic Complexity: 7
Number Of Loops: 0
Number Of Parameters: 1
```

- **UserPersonalInformationSetUpModel**: equals(Object) - We have identified that this particular method has a high cyclomatic complexity of 11, whereas the regular range is [0,3).



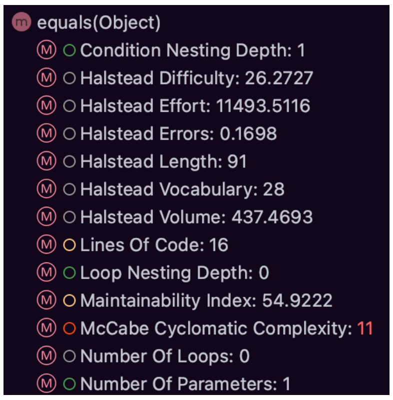
```
equals(Object)
  Condition Nesting Depth: 1
  Halstead Difficulty: 24.7273
  Halstead Effort: 10581.786
  Halstead Errors: 0.1607
  Halstead Length: 90
  Halstead Vocabulary: 27
  Halstead Volume: 427.9399
  Lines Of Code: 16
  Loop Nesting Depth: 0
  Maintainability Index: 54.9926
  McCabe Cyclomatic Complexity: 11
  Number Of Loops: 0
  Number Of Parameters: 1
```

- **UserRequest**: equals(Object) - We have identified that this particular method has a high cyclomatic complexity of 11, whereas the regular range is [0,3).



```
checkIfValid()
  Condition Nesting Depth: 0
  Halstead Difficulty: 17.1
  Halstead Effort: 3452.6423
  Halstead Errors: 0.0761
  Halstead Length: 42
  Halstead Vocabulary: 28
  Halstead Volume: 201.9089
  Lines Of Code: 20
  Loop Nesting Depth: 0
  Maintainability Index: 55.1968
  McCabe Cyclomatic Complexity: 9
  Number Of Loops: 0
  Number Of Parameters: 0
```

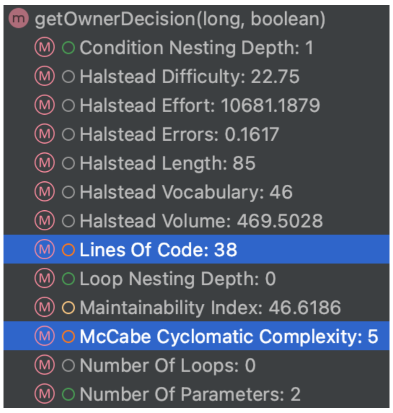
- **UserRequest**: checkIfValid() - We have identified that this particular method has a high cyclomatic complexity of 9, whereas the regular range is [0,3). Here, we decided not to change the code in order to preserve its functionality, since it is one of the most important parts of this microservice.



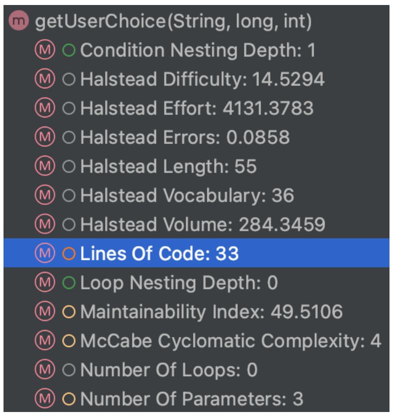
```
equals(Object)
  Condition Nesting Depth: 1
  Halstead Difficulty: 26.2727
  Halstead Effort: 11493.5116
  Halstead Errors: 0.1698
  Halstead Length: 91
  Halstead Vocabulary: 28
  Halstead Volume: 437.4693
  Lines Of Code: 16
  Loop Nesting Depth: 0
  Maintainability Index: 54.9222
  McCabe Cyclomatic Complexity: 11
  Number Of Loops: 0
  Number Of Parameters: 1
```

1.1.2 Scheduler Microservice

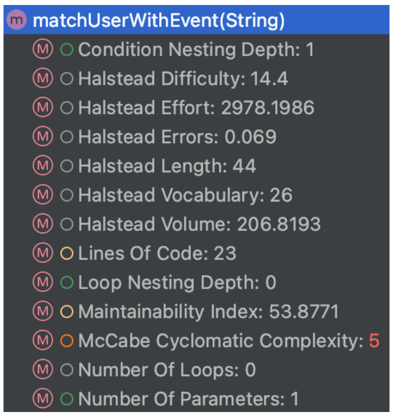
- **SchedulerController**: getOwnerDecision() - We have identified that this method has a high cyclomatic complexity of 5, whereas the regular range is [0, 3). Also, this method has too many lines of code, 38, while the regular range is [0, 11).


```
getOwnerDecision(long, boolean)
    Condition Nesting Depth: 1
    Halstead Difficulty: 22.75
    Halstead Effort: 10681.1879
    Halstead Errors: 0.1617
    Halstead Length: 85
    Halstead Vocabulary: 46
    Halstead Volume: 469.5028
    Lines Of Code: 38
    Loop Nesting Depth: 0
    Maintainability Index: 46.6186
    McCabe Cyclomatic Complexity: 5
    Number Of Loops: 0
    Number Of Parameters: 2
```

- **SchedulerController**: getUserChoice() - We have identified that this method has too many lines of code, 33, whereas the regular range is [0, 11).


```
getUserChoice(String, long, int)
    Condition Nesting Depth: 1
    Halstead Difficulty: 14.5294
    Halstead Effort: 4131.3783
    Halstead Errors: 0.0858
    Halstead Length: 55
    Halstead Vocabulary: 36
    Halstead Volume: 284.3459
    Lines Of Code: 33
    Loop Nesting Depth: 0
    Maintainability Index: 49.5106
    McCabe Cyclomatic Complexity: 4
    Number Of Loops: 0
    Number Of Parameters: 3
```

- **SchedulerController**: matchUserWithEvent() - We have identified that this method has a high cyclomatic complexity of 5, whereas the regular range is [0, 3)

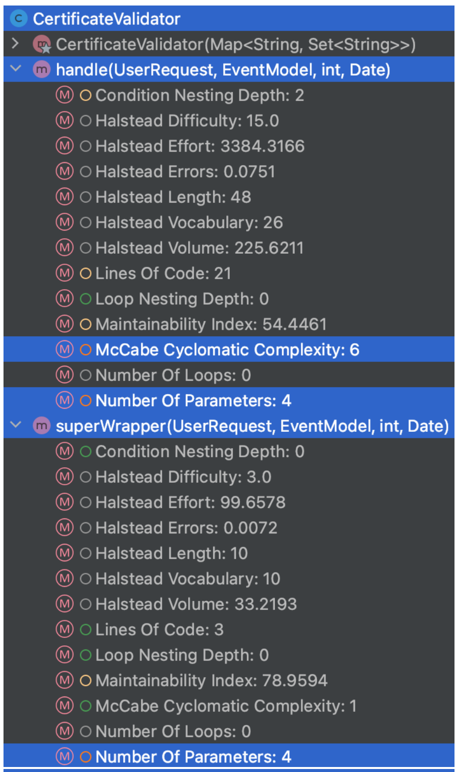

```
matchUserWithEvent(String)
    Condition Nesting Depth: 1
    Halstead Difficulty: 14.4
    Halstead Effort: 2978.1986
    Halstead Errors: 0.069
    Halstead Length: 44
    Halstead Vocabulary: 26
    Halstead Volume: 206.8193
    Lines Of Code: 23
    Loop Nesting Depth: 0
    Maintainability Index: 53.8771
    McCabe Cyclomatic Complexity: 5
    Number Of Loops: 0
    Number Of Parameters: 1
```

- **Validator interface**: handle() - We have identified that this method from the interface has a high number of parameters 4, while the regular range is [0, 3). As a consequence, all the classes that implement this interface have a high number of parameters in this method. They are CompetitionValidator, CertificateValidator, and AvailabilityValidator.
- **BaseValidator**: checkNext() - We have identified that this method has a high number of parameters, 4, while the regular range is [0, 3).
- **CompetitionValidator**: superWrapper() - We have identified that this method has a high number of parameters, 4, while the regular range is [0, 3)

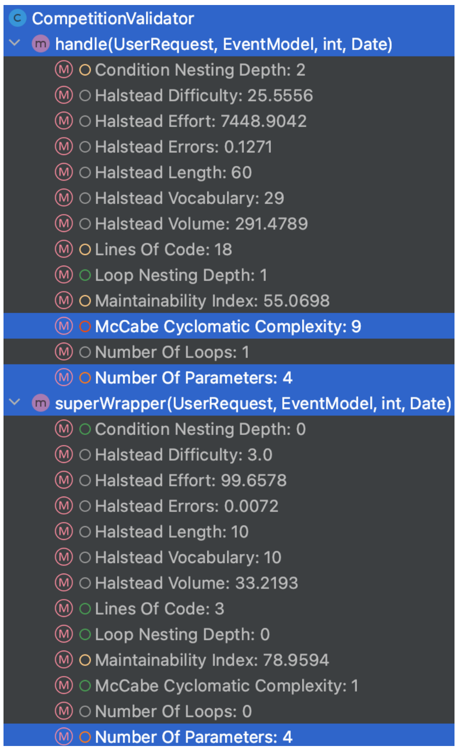
- **CertificateValidator**: superWrapper() - We have identified that this method has a high number of parameters, 4, while the regular range is [0, 3)

- **Availability**: superWrapper() - We have identified that this method has a high number of parameters, 4, while the regular range is [0, 3)

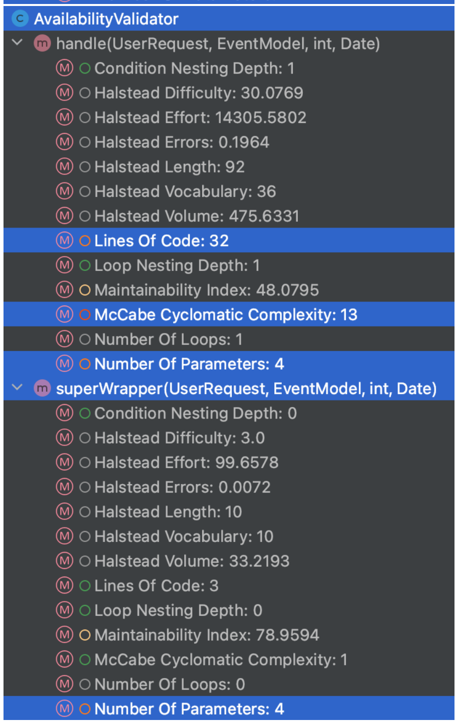
- **Availability**: handle() - We have identified that this method has a high cyclomatic complexity of 13, while the regular range is [0, 3). Also, the number of lines of code is too high, 32, while the regular range is [0, 11)



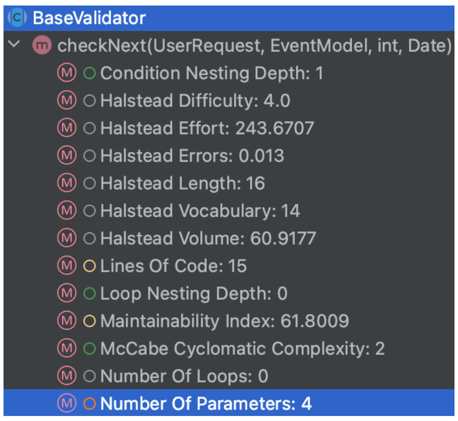
```
CertificateValidator
  > CertificateValidator(Map<String, Set<String>>)
  v handle(UserRequest, EventModel, int, Date)
      Condition Nesting Depth: 2
      Halstead Difficulty: 15.0
      Halstead Effort: 3384.3166
      Halstead Errors: 0.0751
      Halstead Length: 48
      Halstead Vocabulary: 26
      Halstead Volume: 225.6211
      Lines Of Code: 21
      Loop Nesting Depth: 0
      Maintainability Index: 54.4461
      McCabe Cyclomatic Complexity: 6
      Number Of Loops: 0
      Number Of Parameters: 4
  v superWrapper(UserRequest, EventModel, int, Date)
      Condition Nesting Depth: 0
      Halstead Difficulty: 3.0
      Halstead Effort: 99.6578
      Halstead Errors: 0.0072
      Halstead Length: 10
      Halstead Vocabulary: 10
      Halstead Volume: 33.2193
      Lines Of Code: 3
      Loop Nesting Depth: 0
      Maintainability Index: 78.9594
      McCabe Cyclomatic Complexity: 1
      Number Of Loops: 0
      Number Of Parameters: 4
```



```
CompetitionValidator
  v handle(UserRequest, EventModel, int, Date)
      Condition Nesting Depth: 2
      Halstead Difficulty: 25.5556
      Halstead Effort: 7448.9042
      Halstead Errors: 0.1271
      Halstead Length: 60
      Halstead Vocabulary: 29
      Halstead Volume: 291.4789
      Lines Of Code: 18
      Loop Nesting Depth: 1
      Maintainability Index: 55.0698
      McCabe Cyclomatic Complexity: 9
      Number Of Loops: 1
      Number Of Parameters: 4
  v superWrapper(UserRequest, EventModel, int, Date)
      Condition Nesting Depth: 0
      Halstead Difficulty: 3.0
      Halstead Effort: 99.6578
      Halstead Errors: 0.0072
      Halstead Length: 10
      Halstead Vocabulary: 10
      Halstead Volume: 33.2193
      Lines Of Code: 3
      Loop Nesting Depth: 0
      Maintainability Index: 78.9594
      McCabe Cyclomatic Complexity: 1
      Number Of Loops: 0
      Number Of Parameters: 4
```



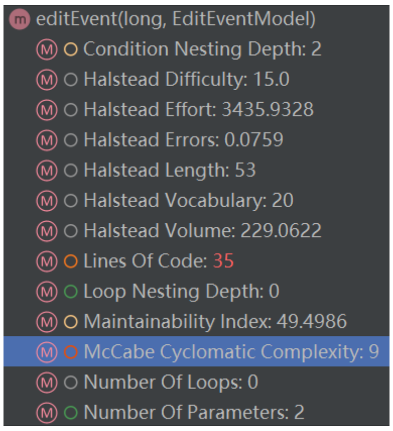
```
AvailabilityValidator
  v handle(UserRequest, EventModel, int, Date)
      Condition Nesting Depth: 1
      Halstead Difficulty: 30.0769
      Halstead Effort: 14305.5802
      Halstead Errors: 0.1964
      Halstead Length: 92
      Halstead Vocabulary: 36
      Halstead Volume: 475.6331
      Lines Of Code: 32
      Loop Nesting Depth: 1
      Maintainability Index: 48.0795
      McCabe Cyclomatic Complexity: 13
      Number Of Loops: 1
      Number Of Parameters: 4
  v superWrapper(UserRequest, EventModel, int, Date)
      Condition Nesting Depth: 0
      Halstead Difficulty: 3.0
      Halstead Effort: 99.6578
      Halstead Errors: 0.0072
      Halstead Length: 10
      Halstead Vocabulary: 10
      Halstead Volume: 33.2193
      Lines Of Code: 3
      Loop Nesting Depth: 0
      Maintainability Index: 78.9594
      McCabe Cyclomatic Complexity: 1
      Number Of Loops: 0
      Number Of Parameters: 4
```



```
BaseValidator
  v checkNext(UserRequest, EventModel, int, Date)
      Condition Nesting Depth: 1
      Halstead Difficulty: 4.0
      Halstead Effort: 243.6707
      Halstead Errors: 0.013
      Halstead Length: 16
      Halstead Vocabulary: 14
      Halstead Volume: 60.9177
      Lines Of Code: 15
      Loop Nesting Depth: 0
      Maintainability Index: 61.8009
      McCabe Cyclomatic Complexity: 2
      Number Of Loops: 0
      Number Of Parameters: 4
```

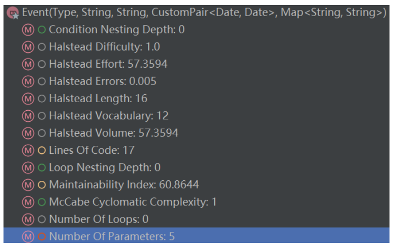
1.1.3 Event Microservice

- **EventService**: editEvent() - We have identified that this particular method has a high cyclomatic complexity of 9, whereas the regular range is [0, 3).



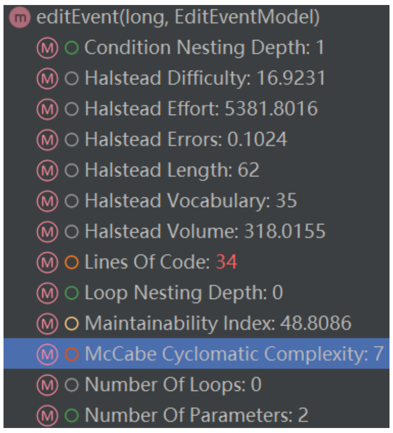
```
editEvent(long, EditEventModel)
    Condition Nesting Depth: 2
    Halstead Difficulty: 15.0
    Halstead Effort: 3435.9328
    Halstead Errors: 0.0759
    Halstead Length: 53
    Halstead Vocabulary: 20
    Halstead Volume: 229.0622
    Lines Of Code: 35
    Loop Nesting Depth: 0
    Maintainability Index: 49.4986
    McCabe Cyclomatic Complexity: 9
    Number Of Loops: 0
    Number Of Parameters: 2
```

- **Event**: constructor - We have identified that this constructor has a high number of parameters of 5, whereas the regular range is [0, 3).



```
Event(Type, String, String, CustomPair<Date, Date>, Map<String, String>)
    Condition Nesting Depth: 0
    Halstead Difficulty: 1.0
    Halstead Effort: 57.3594
    Halstead Errors: 0.005
    Halstead Length: 16
    Halstead Vocabulary: 12
    Halstead Volume: 57.3594
    Lines Of Code: 17
    Loop Nesting Depth: 0
    Maintainability Index: 60.8644
    McCabe Cyclomatic Complexity: 1
    Number Of Loops: 0
    Number Of Parameters: 5
```

- **EventController**: editEvent() - We have identified that this particular method has a high cyclomatic complexity of 7, whereas the regular range is [0, 3).

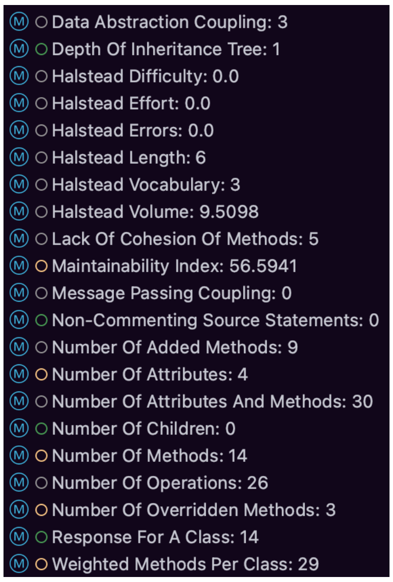


```
editEvent(long, EditEventModel)
    Condition Nesting Depth: 1
    Halstead Difficulty: 16.9231
    Halstead Effort: 5381.8016
    Halstead Errors: 0.1024
    Halstead Length: 62
    Halstead Vocabulary: 35
    Halstead Volume: 318.0155
    Lines Of Code: 34
    Loop Nesting Depth: 0
    Maintainability Index: 48.8086
    McCabe Cyclomatic Complexity: 7
    Number Of Loops: 0
    Number Of Parameters: 2
```

1.2 Classes

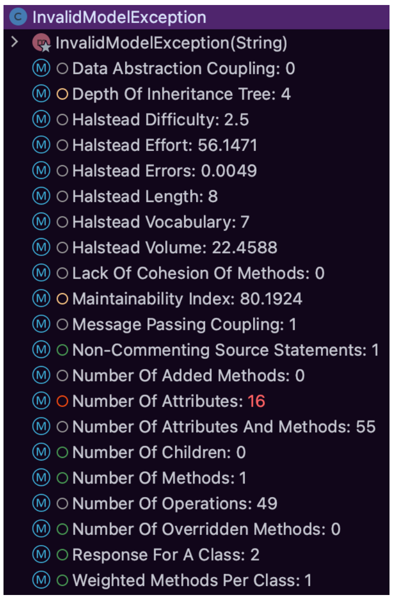
1.2.1 User Microservice

- **EventModel**: The event model class has a larger number of attributes (4) and number of methods (14). The number of methods will be fixed through refactoring, but we decided to stick with the same number of attributes since that is also one of the core components in our project and is needed in all microservices.



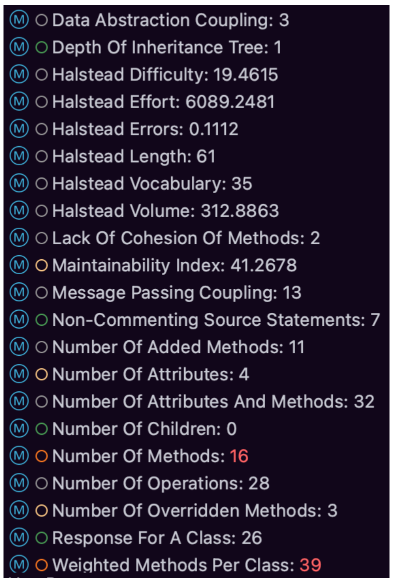
```
Data Abstraction Coupling: 3
Depth Of Inheritance Tree: 1
Halstead Difficulty: 0.0
Halstead Effort: 0.0
Halstead Errors: 0.0
Halstead Length: 6
Halstead Vocabulary: 3
Halstead Volume: 9.5098
Lack Of Cohesion Of Methods: 5
Maintainability Index: 56.5941
Message Passing Coupling: 0
Non-Commenting Source Statements: 0
Number Of Added Methods: 9
Number Of Attributes: 4
Number Of Attributes And Methods: 30
Number Of Children: 0
Number Of Methods: 14
Number Of Operations: 26
Number Of Overridden Methods: 3
Response For A Class: 14
Weighted Methods Per Class: 29
```

- **InvalidModelException**: The InvalidModelException class has a high 16 attributes, which is not within the regular range of [0,4). Regardless, that is because the class extends the RunTimeException class and it inherits its attributes, so we decided to keep it as it is.



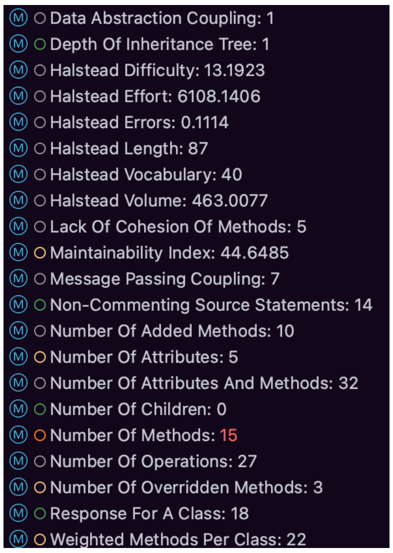
```
InvalidModelException
> InvalidModelException(String)
    Data Abstraction Coupling: 0
    Depth Of Inheritance Tree: 4
    Halstead Difficulty: 2.5
    Halstead Effort: 56.1471
    Halstead Errors: 0.0049
    Halstead Length: 8
    Halstead Vocabulary: 7
    Halstead Volume: 22.4588
    Lack Of Cohesion Of Methods: 0
    Maintainability Index: 80.1924
    Message Passing Coupling: 1
    Non-Commenting Source Statements: 1
    Number Of Added Methods: 0
    Number Of Attributes: 16
    Number Of Attributes And Methods: 55
    Number Of Children: 0
    Number Of Methods: 1
    Number Of Operations: 49
    Number Of Overridden Methods: 0
    Response For A Class: 2
    Weighted Methods Per Class: 1
```

- **UserRequest**: The UserRequest class has many attributes (4), too many methods (16), and high weighted methods per class value (39). The last two issues will be fixed through refactoring, whereas the first one will remain present in our code since the attributes of the UserRequest are all maintaining our project's functionality.



```
Ⓜ ○ Data Abstraction Coupling: 3
Ⓜ ○ Depth Of Inheritance Tree: 1
Ⓜ ○ Halstead Difficulty: 19.4615
Ⓜ ○ Halstead Effort: 6089.2481
Ⓜ ○ Halstead Errors: 0.1112
Ⓜ ○ Halstead Length: 61
Ⓜ ○ Halstead Vocabulary: 35
Ⓜ ○ Halstead Volume: 312.8863
Ⓜ ○ Lack Of Cohesion Of Methods: 2
Ⓜ ○ Maintainability Index: 41.2678
Ⓜ ○ Message Passing Coupling: 13
Ⓜ ○ Non-Commenting Source Statements: 7
Ⓜ ○ Number Of Added Methods: 11
Ⓜ ○ Number Of Attributes: 4
Ⓜ ○ Number Of Attributes And Methods: 32
Ⓜ ○ Number Of Children: 0
Ⓜ ○ Number Of Methods: 16
Ⓜ ○ Number Of Operations: 28
Ⓜ ○ Number Of Overridden Methods: 3
Ⓜ ○ Response For A Class: 26
Ⓜ ○ Weighted Methods Per Class: 39
```

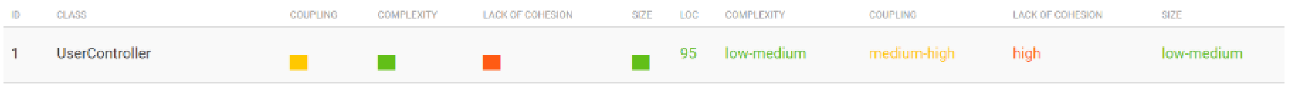
- **User**: The User class has a high number of methods (15) which does not fall into the regular range of [0,15). Refactoring does not help in this case, since all 15 methods are used inside the project, so we also decided to stick to our initial structure.



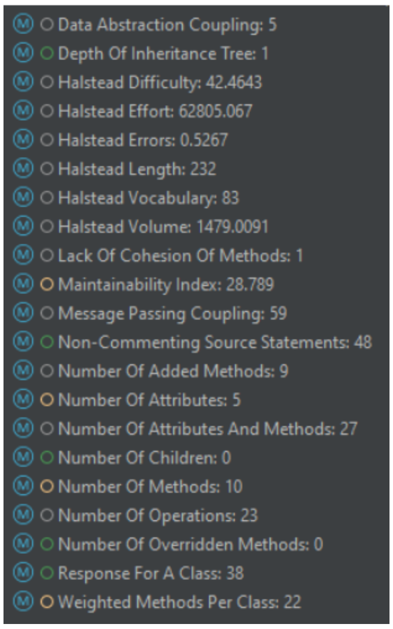
```
Ⓜ ○ Data Abstraction Coupling: 1
Ⓜ ○ Depth Of Inheritance Tree: 1
Ⓜ ○ Halstead Difficulty: 13.1923
Ⓜ ○ Halstead Effort: 6108.1406
Ⓜ ○ Halstead Errors: 0.1114
Ⓜ ○ Halstead Length: 87
Ⓜ ○ Halstead Vocabulary: 40
Ⓜ ○ Halstead Volume: 463.0077
Ⓜ ○ Lack Of Cohesion Of Methods: 5
Ⓜ ○ Maintainability Index: 44.6485
Ⓜ ○ Message Passing Coupling: 7
Ⓜ ○ Non-Commenting Source Statements: 14
Ⓜ ○ Number Of Added Methods: 10
Ⓜ ○ Number Of Attributes: 5
Ⓜ ○ Number Of Attributes And Methods: 32
Ⓜ ○ Number Of Children: 0
Ⓜ ○ Number Of Methods: 15
Ⓜ ○ Number Of Operations: 27
Ⓜ ○ Number Of Overridden Methods: 3
Ⓜ ○ Response For A Class: 18
Ⓜ ○ Weighted Methods Per Class: 22
```

- **UserController**: The UserController class was the only one labeled as high (red) LCOM (Lack of Cohesion of Methods) in the initial CodeMR report. Therefore, it was of the utmost importance to refactor this class. Upon a more in-depth analysis using TreeMetrics, we noticed that the following metrics were also too high:
  - NOA (Number of Attributes) = 5, while the regular range is [0, 4)
  - NOM (Number of Methods) = 10, while the regular range is [0, 7)
  - WMC (Weighted Method Count) = 22, while the regular range is [0, 12)

  Although the NOA is not far from the limits, the number of methods seems to be quite out of bounds, as well as the WMC.



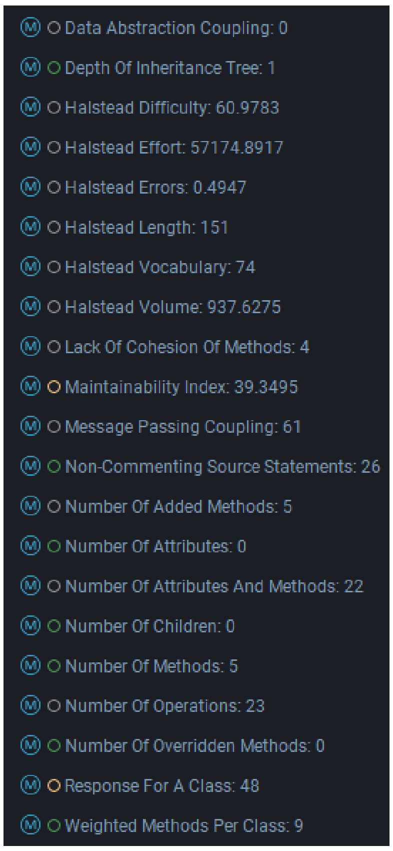
ID	CLASS	COUPLING	COMPLEXITY	LACK OF COHESION	SIZE	LOC	COMPLEXITY	COUPLING	LACK OF COHESION	SIZE
1	UserController	🟨	🟩	🟥	🟩	95	low-medium	medium-high	high	low-medium



```
Data Abstraction Coupling: 5
Depth Of Inheritance Tree: 1
Halstead Difficulty: 42.4643
Halstead Effort: 62805.067
Halstead Errors: 0.5267
Halstead Length: 232
Halstead Vocabulary: 83
Halstead Volume: 1479.0091
Lack Of Cohesion Of Methods: 1
Maintainability Index: 28.789
Message Passing Coupling: 59
Non-Commenting Source Statements: 48
Number Of Added Methods: 9
Number Of Attributes: 5
Number Of Attributes And Methods: 27
Number Of Children: 0
Number Of Methods: 10
Number Of Operations: 23
Number Of Overridden Methods: 0
Response For A Class: 38
Weighted Methods Per Class: 22
```

- **UserService/UserServiceImpl**: The UserService and the respective UserServiceImpl had been labeled as medium-high (yellow) coupling in the initial CodeMR report. When analyzing the class further via TreeMetrics, we also found another metric to be high: the RFAC (Response for a Class) was identified with a value of 48, while a normal range would be [0, 45).

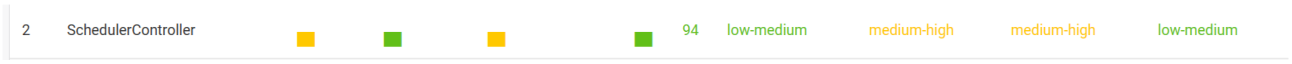
2	UserServiceImpl	🟨	🟩	🟩	🟩	55	low	medium-high	low	low-medium



```
Data Abstraction Coupling: 0
Depth Of Inheritance Tree: 1
Halstead Difficulty: 60.9783
Halstead Effort: 57174.8917
Halstead Errors: 0.4947
Halstead Length: 151
Halstead Vocabulary: 74
Halstead Volume: 937.6275
Lack Of Cohesion Of Methods: 4
Maintainability Index: 39.3495
Message Passing Coupling: 61
Non-Commenting Source Statements: 26
Number Of Added Methods: 5
Number Of Attributes: 0
Number Of Attributes And Methods: 22
Number Of Children: 0
Number Of Methods: 5
Number Of Operations: 23
Number Of Overridden Methods: 0
Response For A Class: 48
Weighted Methods Per Class: 9
```

1.2.2 Scheduler Microservice

- **SchedulerController**: According to CodeMR, the SchedulerController class was considered to have a medium-high (yellow) LCOM. Although this metric was not necessarily worrying, since this controller manages a crucial part of our functionality, we decided to improve its maintainability by refactoring it. After we had refactored and improved the LOCM, as shown in task 2, we took notice of the fact that we also had medium-high (yellow) coupling for this class. We, therefore, decided to refactor the class even more, which is also described in task 2, to further improve the metrics.

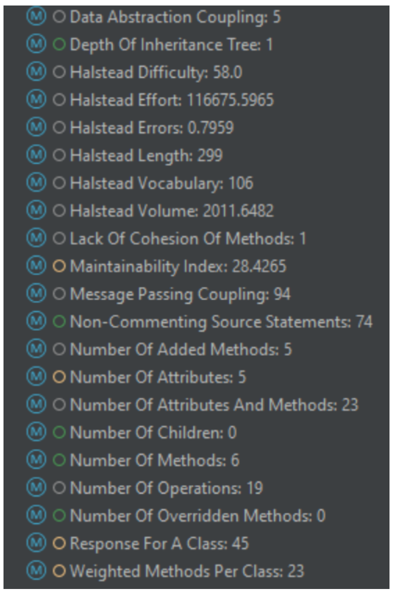


2	SchedulerController	🟨	🟩	🟨	🟩	94	low-medium	medium-high	medium-high	low-medium

Upon a more in-depth analysis using TreeMetrics, we found that there are some more metrics that are too high:

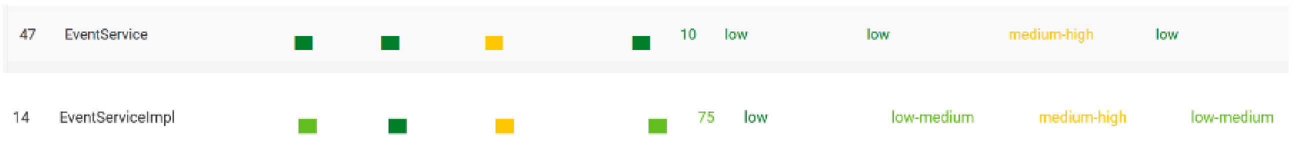
  - NOA = 5, while the regular range is [0,4)
  - RFC = 45, while the regular range is [0,45)
  - WMC = 23, while the regular range is [0,12)

Although the NOA and RFC are not far from the limits, the WMC seems to be an issue.



```
Data Abstraction Coupling: 5
Depth Of Inheritance Tree: 1
Halstead Difficulty: 58.0
Halstead Effort: 116675.5965
Halstead Errors: 0.7959
Halstead Length: 299
Halstead Vocabulary: 106
Halstead Volume: 2011.6482
Lack Of Cohesion Of Methods: 1
Maintainability Index: 28.4265
Message Passing Coupling: 94
Non-Commenting Source Statements: 74
Number Of Added Methods: 5
Number Of Attributes: 5
Number Of Attributes And Methods: 23
Number Of Children: 0
Number Of Methods: 6
Number Of Operations: 19
Number Of Overridden Methods: 0
Response For A Class: 45
Weighted Methods Per Class: 23
```

- **EventService/EventServiceImpl**: The EventService and the respective EventServiceImpl had been labeled with medium-high (yellow) LCOM in the initial CodeMR report.

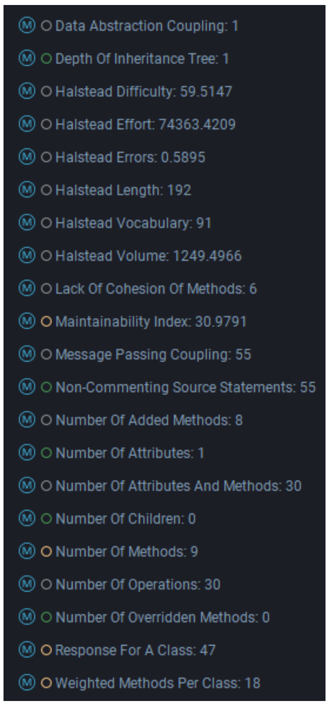


47	EventService	🟩	🟩	🟨	🟩	10	low	low	medium-high	low
14	EventServiceImpl	🟩	🟩	🟨	🟩	75	low	low-medium	medium-high	low-medium

When running TreeMetrics on this service, we also found a multitude of other too-high metrics:

  - NOM = 9, while the regular range is [0, 7)

- RFAC = 47, while the regular range is [0, 45)
- WMC = 18, while the regular range is [0, 12)



```
Data Abstraction Coupling: 1
Depth Of Inheritance Tree: 1
Halstead Difficulty: 59.5147
Halstead Effort: 74363.4209
Halstead Errors: 0.5895
Halstead Length: 192
Halstead Vocabulary: 91
Halstead Volume: 1249.4966
Lack Of Cohesion Of Methods: 6
Maintainability Index: 30.9791
Message Passing Coupling: 55
Non-Commenting Source Statements: 55
Number Of Added Methods: 8
Number Of Attributes: 1
Number Of Attributes And Methods: 30
Number Of Children: 0
Number Of Methods: 9
Number Of Operations: 30
Number Of Overridden Methods: 0
Response For A Class: 47
Weighted Methods Per Class: 18
```

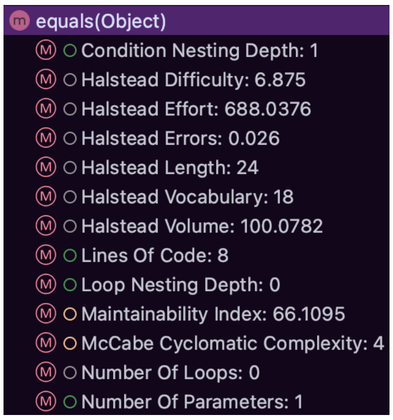
Task 2: Code Refactoring

2.1 Methods

2.1.1 User Microservice

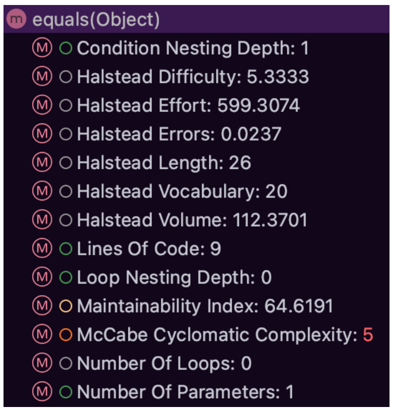
- **EventIdModel**: equals(Object) - Since there was no use of the equals() method, we decided to remove it from the code.

- **CustomPair**: equals(Object) - We removed a part of the equals method that was unnecessary in order to lower the cyclomatic complexity.



```
equals(Object)
Condition Nesting Depth: 1
Halstead Difficulty: 6.875
Halstead Effort: 688.0376
Halstead Errors: 0.026
Halstead Length: 24
Halstead Vocabulary: 18
Halstead Volume: 100.0782
Lines Of Code: 8
Loop Nesting Depth: 0
Maintainability Index: 66.1095
McCabe Cyclomatic Complexity: 4
Number Of Loops: 0
Number Of Parameters: 1
```

- **EventModel**: equals(Object) - We removed a part of the equals method that was unnecessary in order to lower the cyclomatic complexity.



```
equals(Object)
  Condition Nesting Depth: 1
  Halstead Difficulty: 5.3333
  Halstead Effort: 599.3074
  Halstead Errors: 0.0237
  Halstead Length: 26
  Halstead Vocabulary: 20
  Halstead Volume: 112.3701
  Lines Of Code: 9
  Loop Nesting Depth: 0
  Maintainability Index: 64.6191
  McCabe Cyclomatic Complexity: 5
  Number Of Loops: 0
  Number Of Parameters: 1
```

- **UserApprovalModel**: equals(Object) - Since there was no use of the equals() method, we decided to remove it from the code.

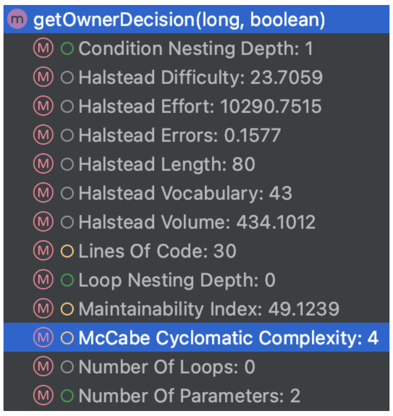
- **UserInformationModel**: equals(Object) - Since there was no use of the equals() method, we decided to remove it from the code.

- **UserPersonalInformationSetUpModel**: equals(Object) - Since there was no use of the equals() method, we decided to remove it from the code.

- **UserRequest**: equals(Object) - Since there was no use of the equals() method, we decided to remove it from the code.

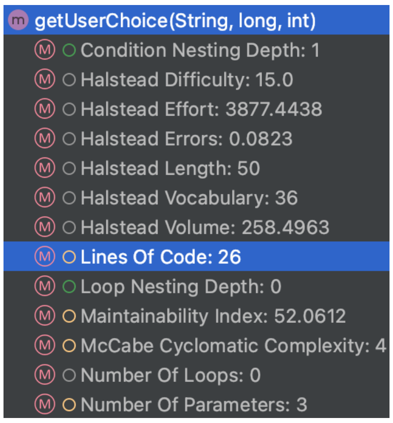
2.1.2 Scheduler Microservice

- **SchedulerDatabaseController**: getOwnerDecision() - This method had a high cyclomatic complexity of 5 and a high number of lines of code, 38. After the refactoring, the new cyclomatic complexity is 4 and it has 30 lines of code. Also, this method was moved from SchedulerController to SchedulerDatabaseController.



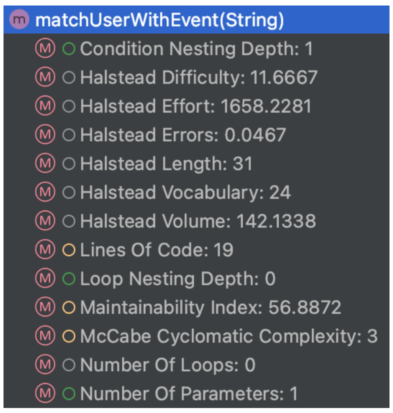
```
getOwnerDecision(long, boolean)
  Condition Nesting Depth: 1
  Halstead Difficulty: 23.7059
  Halstead Effort: 10290.7515
  Halstead Errors: 0.1577
  Halstead Length: 80
  Halstead Vocabulary: 43
  Halstead Volume: 434.1012
  Lines Of Code: 30
  Loop Nesting Depth: 0
  Maintainability Index: 49.1239
  McCabe Cyclomatic Complexity: 4
  Number Of Loops: 0
  Number Of Parameters: 2
```

- **SchedulerDatabaseController**: getUserChoice() - This method had a high number of lines of code, 33. After refactoring, we decreased this to 26. Also, this method was moved from SchedulerController to SchedulerDatabaseController.



```
getUserChoice(String, long, int)
  Condition Nesting Depth: 1
  Halstead Difficulty: 15.0
  Halstead Effort: 3877.4438
  Halstead Errors: 0.0823
  Halstead Length: 50
  Halstead Vocabulary: 36
  Halstead Volume: 258.4963
  Lines Of Code: 26
  Loop Nesting Depth: 0
  Maintainability Index: 52.0612
  McCabe Cyclomatic Complexity: 4
  Number Of Loops: 0
  Number Of Parameters: 3
```

- **SchedulerController**: matchUserWithEvent() - This method had a high cyclomatic complexity of 5. After refactoring, this number decreased to 3.



```
matchUserWithEvent(String)
  Condition Nesting Depth: 1
  Halstead Difficulty: 11.6667
  Halstead Effort: 1658.2281
  Halstead Errors: 0.0467
  Halstead Length: 31
  Halstead Vocabulary: 24
  Halstead Volume: 142.1338
  Lines Of Code: 19
  Loop Nesting Depth: 0
  Maintainability Index: 56.8872
  McCabe Cyclomatic Complexity: 3
  Number Of Loops: 0
  Number Of Parameters: 1
```

- **Validator interface**: handle() - This method from the interface had a high number of parameters, 4. After the refactoring, this number decreased to 2. As a consequence, in all the classes that implement this interface(CompetitionValidator, CertificateValidator and AvailabilityValidator), the number of parameters from this method decreases.

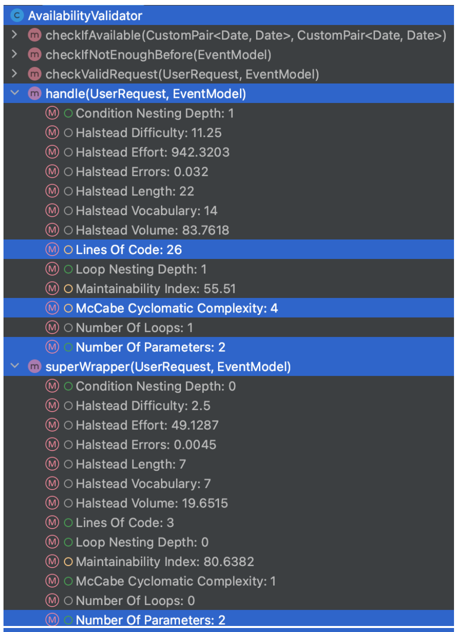
- **BaseValidator**: checkNext() - This method had a high number of parameters, 4. After refactoring, this number decreased to 2.

- **CompetitionValidator**: superWrapper() - This method had a high number of parameters, 4. After refactoring, this number decreased to 2.

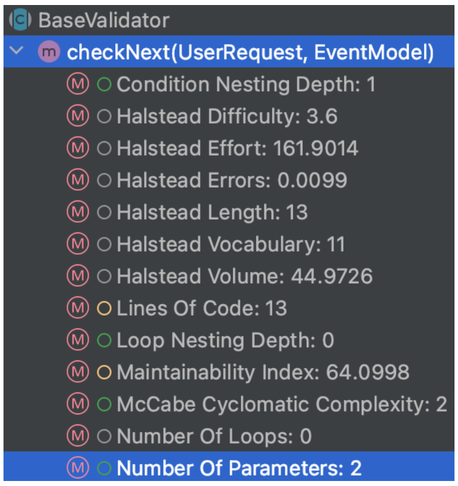
- **CertificateValidator**: superWrapper() - This method had a high number of parameters, 4. After refactoring, this number decreased to 2.

- **Availability**: superWrapper() - This method had a high number of parameters, 4. After refactoring, this number decreased to 2.

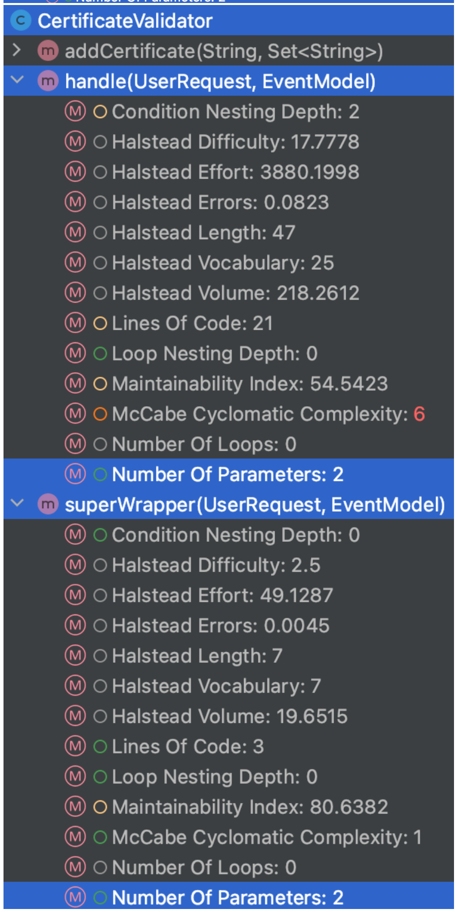
- **Availability**: handle() - This method had a high cyclomatic complexity of 13 and 32 lines of code. After refactoring cyclomatic complexity decreased to 4 and the lines of code decreased to 26.



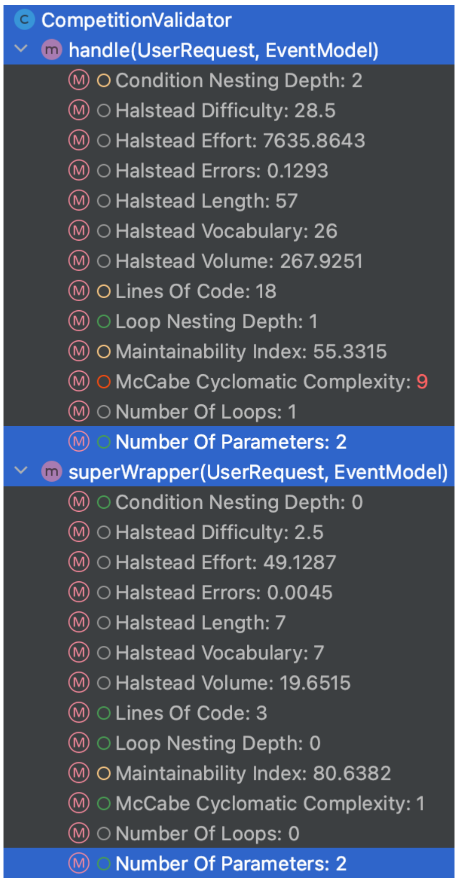
```
AvailabilityValidator
> checkIfAvailable(CustomPair<Date, Date>, CustomPair<Date, Date>)
> checkIfNotEnoughBefore(EventModel)
> checkValidRequest(UserRequest, EventModel)
v handle(UserRequest, EventModel)
    Condition Nesting Depth: 1
    Halstead Difficulty: 11.25
    Halstead Effort: 942.3203
    Halstead Errors: 0.032
    Halstead Length: 22
    Halstead Vocabulary: 14
    Halstead Volume: 83.7618
    Lines Of Code: 26
    Loop Nesting Depth: 1
    Maintainability Index: 55.51
    McCabe Cyclomatic Complexity: 4
    Number Of Loops: 1
    Number Of Parameters: 2
v superWrapper(UserRequest, EventModel)
    Condition Nesting Depth: 0
    Halstead Difficulty: 2.5
    Halstead Effort: 49.1287
    Halstead Errors: 0.0045
    Halstead Length: 7
    Halstead Vocabulary: 7
    Halstead Volume: 19.6515
    Lines Of Code: 3
    Loop Nesting Depth: 0
    Maintainability Index: 80.6382
    McCabe Cyclomatic Complexity: 1
    Number Of Loops: 0
    Number Of Parameters: 2
```



```
BaseValidator
v checkNext(UserRequest, EventModel)
    Condition Nesting Depth: 1
    Halstead Difficulty: 3.6
    Halstead Effort: 161.9014
    Halstead Errors: 0.0099
    Halstead Length: 13
    Halstead Vocabulary: 11
    Halstead Volume: 44.9726
    Lines Of Code: 13
    Loop Nesting Depth: 0
    Maintainability Index: 64.0998
    McCabe Cyclomatic Complexity: 2
    Number Of Loops: 0
    Number Of Parameters: 2
```



```
CertificateValidator
> addCertificate(String, Set<String>)
v handle(UserRequest, EventModel)
    Condition Nesting Depth: 2
    Halstead Difficulty: 17.7778
    Halstead Effort: 3880.1998
    Halstead Errors: 0.0823
    Halstead Length: 47
    Halstead Vocabulary: 25
    Halstead Volume: 218.2612
    Lines Of Code: 21
    Loop Nesting Depth: 0
    Maintainability Index: 54.5423
    McCabe Cyclomatic Complexity: 6
    Number Of Loops: 0
    Number Of Parameters: 2
v superWrapper(UserRequest, EventModel)
    Condition Nesting Depth: 0
    Halstead Difficulty: 2.5
    Halstead Effort: 49.1287
    Halstead Errors: 0.0045
    Halstead Length: 7
    Halstead Vocabulary: 7
    Halstead Volume: 19.6515
    Lines Of Code: 3
    Loop Nesting Depth: 0
    Maintainability Index: 80.6382
    McCabe Cyclomatic Complexity: 1
    Number Of Loops: 0
    Number Of Parameters: 2
```



```
CompetitionValidator
v handle(UserRequest, EventModel)
    Condition Nesting Depth: 2
    Halstead Difficulty: 28.5
    Halstead Effort: 7635.8643
    Halstead Errors: 0.1293
    Halstead Length: 57
    Halstead Vocabulary: 26
    Halstead Volume: 267.9251
    Lines Of Code: 18
    Loop Nesting Depth: 1
    Maintainability Index: 55.3315
    McCabe Cyclomatic Complexity: 9
    Number Of Loops: 1
    Number Of Parameters: 2
v superWrapper(UserRequest, EventModel)
    Condition Nesting Depth: 0
    Halstead Difficulty: 2.5
    Halstead Effort: 49.1287
    Halstead Errors: 0.0045
    Halstead Length: 7
    Halstead Vocabulary: 7
    Halstead Volume: 19.6515
    Lines Of Code: 3
    Loop Nesting Depth: 0
    Maintainability Index: 80.6382
    McCabe Cyclomatic Complexity: 1
    Number Of Loops: 0
    Number Of Parameters: 2
```

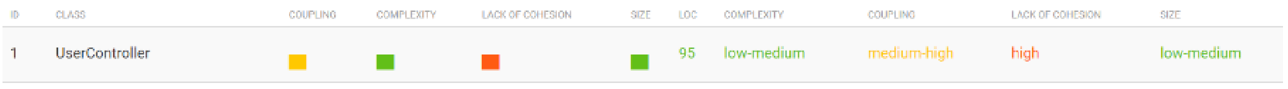
2.2 Classes

2.2.1 User Microservice

We have refactored two separate issues inside the user microservice using the CodeMR data:

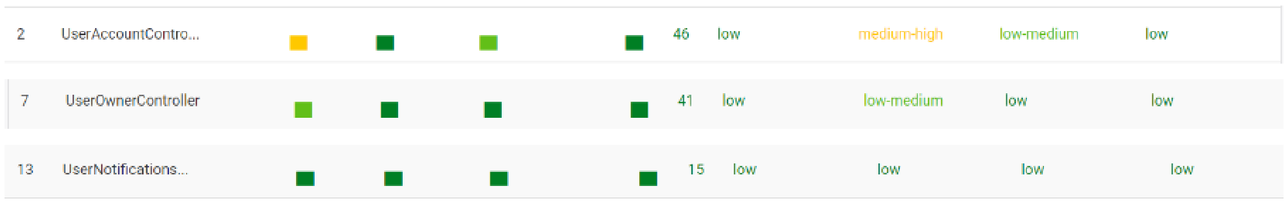
- **UserController**: The UserController had a high lack of cohesion, as stated in the first task. We have resolved this by splitting the UserController into three separate controllers: UserAccountController, UserOwnerController, and UserNotificationsController. This split was based on the different responsibilities each endpoint had and on the services they interact with. First, we have created a UserAccountController, which handles all the functionality related to managing a normal user account and interacts with all user services. Second, we have created a UserOwnerController, which manages the account of a user when it is an owner event and solely interacts with the OwnerService. Finally, we have created a UserNotificationsController which manages the receiving of notifications and communicates only with the NotificationService. This ensures that each of the controllers only performs one functionality, by also accessing only one service. Below, we have added how the metrics look before and after the refactoring.

Before:



ID	CLASS	COUPLING	COMPLEXITY	LACK OF COHESION	SIZE	LOC	COMPLEXITY	COUPLING	LACK OF COHESION	SIZE
1	UserController	🟨	🟩	🟥	🟩	95	low-medium	medium-high	high	low-medium

After:



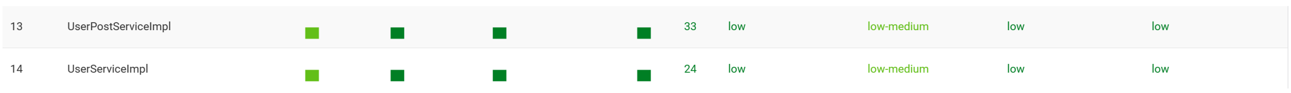
2	UserAccountContro...	🟨	🟩	🟩	🟩	46	low	medium-high	low-medium	low
7	UserOwnerController	🟩	🟩	🟩	🟩	41	low	low-medium	low	low
13	UserNotifications...	🟩	🟩	🟩	🟩	15	low	low	low	low

- **UserService/UserServiceImpl**: The second issue we identified is related to medium-high coupling inside the UserService/UserServiceImpl class. We identified this to be the case because of methods calling each other among other things. We have solved this coupling by creating a new UserPostService and UserPostServiceImpl class. In combination with the already existing service, we have been able to successfully split up the logic between the two classes, which has improved the coupling significantly. Below, we have shown how the metrics look before and after the refactoring.

Before:

2	UserServiceImpl	🟨	🟩	🟩	🟩	55	low	medium-high	low	low-medium

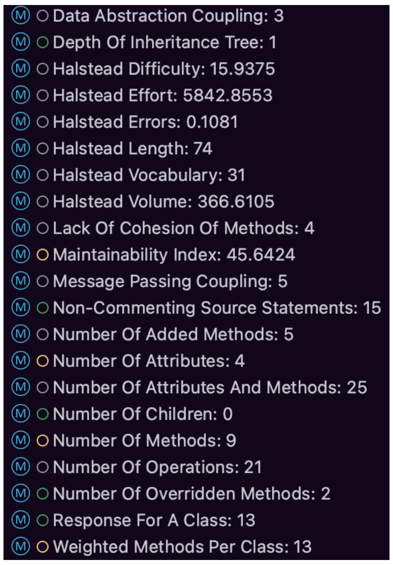
After:



13	UserPostServiceImpl	🟩	🟩	🟩	🟩	33	low	low-medium	low	low
14	UserServiceImpl	🟩	🟩	🟩	🟩	24	low	low-medium	low	low

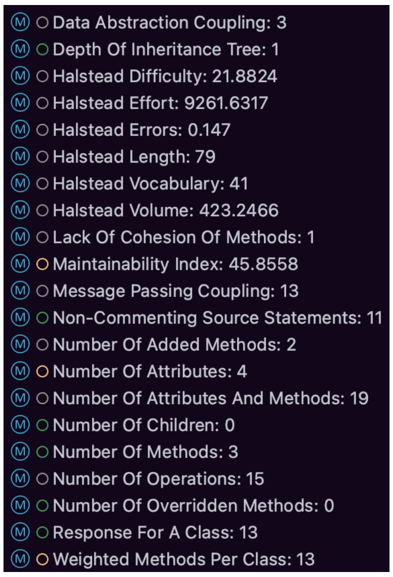
We have also refactored two classes inside the user microservice using the MetricsTree data:

- **EventModel**: Within this class, we decided to quit Lombok annotations because they were automatically generating more redundant methods. Afterwards, we added only the necessary methods.



```
Data Abstraction Coupling: 3
Depth Of Inheritance Tree: 1
Halstead Difficulty: 15.9375
Halstead Effort: 5842.8553
Halstead Errors: 0.1081
Halstead Length: 74
Halstead Vocabulary: 31
Halstead Volume: 366.6105
Lack Of Cohesion Of Methods: 4
Maintainability Index: 45.6424
Message Passing Coupling: 5
Non-Commenting Source Statements: 15
Number Of Added Methods: 5
Number Of Attributes: 4
Number Of Attributes And Methods: 25
Number Of Children: 0
Number Of Methods: 9
Number Of Operations: 21
Number Of Overridden Methods: 2
Response For A Class: 13
Weighted Methods Per Class: 13
```

- **UserRequest**: Within this class, we again decided to quit Lombok annotations because they were automatically generating more redundant methods. Afterwards, we added only the necessary methods.



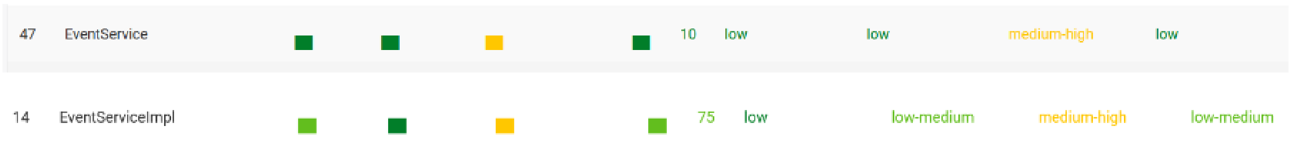
```
Data Abstraction Coupling: 3
Depth Of Inheritance Tree: 1
Halstead Difficulty: 21.8824
Halstead Effort: 9261.6317
Halstead Errors: 0.147
Halstead Length: 79
Halstead Vocabulary: 41
Halstead Volume: 423.2466
Lack Of Cohesion Of Methods: 1
Maintainability Index: 45.8558
Message Passing Coupling: 13
Non-Commenting Source Statements: 11
Number Of Added Methods: 2
Number Of Attributes: 4
Number Of Attributes And Methods: 19
Number Of Children: 0
Number Of Methods: 3
Number Of Operations: 15
Number Of Overridden Methods: 0
Response For A Class: 13
Weighted Methods Per Class: 13
```

2.2.2 Scheduler Microservice

We have refactored three separate issues within the scheduler microservice:

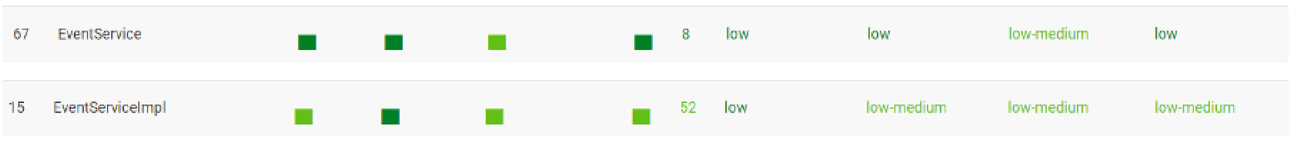
- **EventService/EventServiceImpl**: First, we identified that the EventService and EventServiceImpl classes inside the Scheduler microservice had a medium-high lack of cohesion. We have found this to be the case because a 'certificates' field is only used in one method. We have moved this field into the CertificateValidator class and the addCertificate() method, making them static in the process. This ensures that the EventServiceImpl does not serve multiple functions. Below, we have shown how the metrics changed before and after the refactoring. Before:

Before:



47	EventService	🟩	🟩	🟨	🟩	10	low	low	medium-high	low
14	EventServiceImpl	🟩	🟩	🟨	🟩	75	low	low-medium	medium-high	low-medium

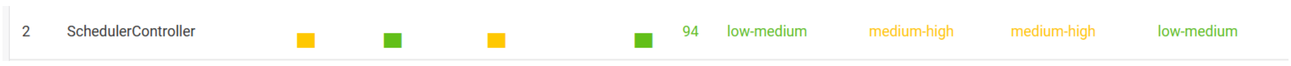
After:



67	EventService	🟩	🟩	🟩	🟩	8	low	low	low-medium	low
15	EventServiceImpl	🟩	🟩	🟩	🟩	52	low	low-medium	low-medium	low-medium

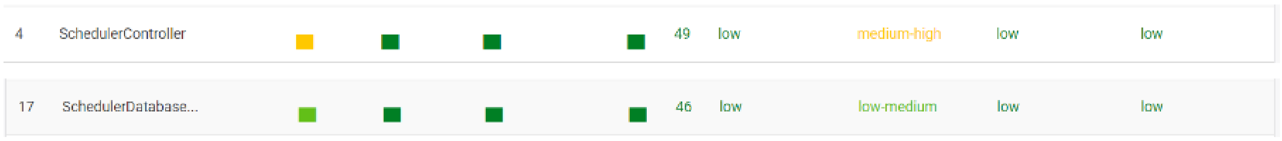
- **SchedulerController**: The second issue we solved is with the SchedulerController. This class also had a medium-high lack of cohesion, as stated before. We have resolved this by splitting the controller into two parts: the previous SchedulerController and a new SchedulerDatabaseController. As the name might suggest, the SchedulerDatabaseController mainly handles the function of endpoints that have to interact with the database, while the existing SchedulerController handles interaction with the EventService. Finally, we have added a CommunicationHandler for the SchedulerDatabaseController, which we use to perform the communication between microservices. This improves cohesion by splitting up functionality and moving methods into a new class. Below, we have shown how the metrics changed before and after the refactoring.

Before:



2	SchedulerController	🟨	🟩	🟨	🟩	94	low-medium	medium-high	medium-high	low-medium

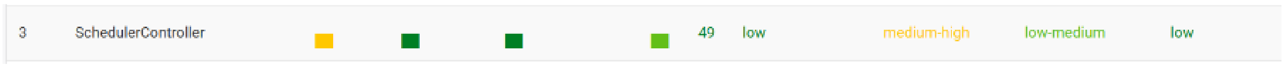
After:



4	SchedulerController	🟨	🟩	🟩	🟩	49	low	medium-high	low	low
17	SchedulerDatabase...	🟩	🟩	🟩	🟩	46	low	low-medium	low	low

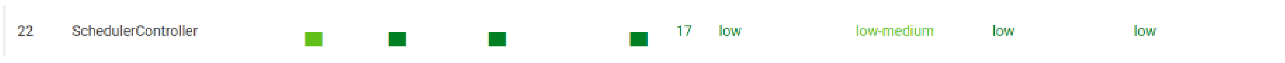
Finally, we improved the medium-high coupling and further improved the low-medium cohesion of the SchedulerController. We have done this by further splitting up the functionality, and adding a new handler that handles the authentication checking: if someone is an admin or not. We have also added two new controllers, a SchedulerCertificateController and a SchedulerNotificationController, which respectively handle the addition of certificates, and the retrieval of notifications. We made sure that the EventService is solely used inside the SchedulerController. With the notification service only being used inside the SchedulerNotificationController.

Before:



3	SchedulerController	🟨	🟩	🟩	🟩	49	low	medium-high	low-medium	low

After:



22	SchedulerController	🟩	🟩	🟩	🟩	17	low	low-medium	low	low