Method level code smells

1. Method Signature: sendRequest(HttpHeaders, Request)

Location: Request microservice -> controllers -> JobRequestController.java
Code Smell:

Metric before:

Method: sendRequest(HttpHeaders, Request)

Metric	Metrics Set	Description	Value	Regular Range
CND		Condition Nesting Depth	4	[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	59	[0..11)
NOPM		Number Of Parameters	2	[0..3)
LAA	Lanza-Mari...	Locality Of Attribute Accesses	0,4286	[0,33..1,0]
FDP	Lanza-Mari...	Foreign Data Providers	2	[0..3)
NOAV	Lanza-Mari...	Number Of Accessed Variables	15	[0..3)
MND	Lanza-Mari...	Maximum Nesting Depth	4	[0..3)
CINT	Lanza-Mari...	Coupling Intensity	22	[0..7)
CDISP	Lanza-Mari...	Coupling Dispersion	0,5652	[0,0..0,5]
HVL	Halstead M...	Halstead Volume	779,1952	
HD	Halstead M...	Halstead Difficulty	27,0	
HL	Halstead M...	Halstead Length	128	
HEF	Halstead M...	Halstead Effort	21038,2716	
HVC	Halstead M...	Halstead Vocabulary	68	
HER	Halstead M...	Halstead Errors	0,254	
MMI	Maintainabi...	Maintainability Index	40,8144	[0,0..19,0]

Action plan:
We want to improve the Cyclomatic Complexity, Lines of Code and Condition Level Nesting. We used the refactoring method called: “Extract Method Refactoring”. The aim is to get both metrics to an “acceptable” level, where ideally LOC is under 40, and CC is the lowest possible, and CND is a regular range where it is equal or less than 2.

The following method was extracted,

```
1 usage   new *
private long getMinutes() {

    LocalDateTime d1 = clockUser.getTimeLdt();
    LocalDate d2 = clockUser.getTimeLd().plusDays( daysToAdd: 1L);
    LocalDateTime ref = d2.atStartOfDay();

    return d1.until(ref, ChronoUnit.MINUTES);
}
```

Method: sendRequest(HttpHeaders, Request)

Metric	Metrics Set	Description	Value	Regular Ra..
CND		Condition Nesting Depth	4	[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	60	[0..11)
NOPM		Number Of Parameters	2	[0..3)
LAA	Lanza-Mari...	Locality Of Attribute Accesses	0,5	[0,33..1,0]
FDP	Lanza-Mari...	Foreign Data Providers	1	[0..3)
NOAV	Lanza-Mari...	Number Of Accessed Variables	12	[0..3)
MND	Lanza-Mari...	Maximum Nesting Depth	4	[0..3)
CINT	Lanza-Mari...	Coupling Intensity	17	[0..7)
CDISP	Lanza-Mari...	Coupling Dispersion	0,5263	[0,0..0,5]
HVL	Halstead M...	Halstead Volume	774,0	
HD	Halstead M...	Halstead Difficulty	27,5	
HL	Halstead M...	Halstead Length	129	
HEF	Halstead M...	Halstead Effort	21285,0	
HVC	Halstead M...	Halstead Vocabulary	64	
HER	Halstead M...	Halstead Errors	0,256	
MMI	Maintainabi...	Maintainability Index	40,6748	[0,0..19,0]

Next, we changed the if-else statements to fix the Condition Nesting Depth and improve the Cyclomatic Complexity.
Since a ResponseEntity is returned in all of the if-else statements, we can also use method extraction here to refactor this method.

The following method was extracted, and this was the new metrics:

```java
private ResponseEntity<String> getResponseEntity(String token, Request request)
        throws JsonProcessingException {

    long minutes = getMinutes();
    List<String> facultyUserFaculties = requestAllocationService.getFacultyUserFaculties(token);

    if (clockUser.getTimeLd().isEqual(request.getPreferredDate())) {
        return ResponseEntity.ok().body("You cannot send requests for the same day.");
    }
    if (minutes <= 5) {
        return ResponseEntity.ok().body("You cannot send requests 5 min before the following day.");
    }
    if (minutes <= 360) {
        if (requestAllocationService.enoughResourcesForJob(request, token)
                && request.getPreferredDate().equals(clockUser.getTimeLd().plusDays( daysToAdd: 1L))
                && (facultyUserFaculties.contains(request.getFaculty()))) {
            request.setApproved(true);                // Doesn't require approval; First come, first served
            requestRepository.save(request);
            publishRequest();
            this.requestAllocationService.sendRequestToCluster(request, token);

            return ResponseEntity.ok()
                    .body("The request is automatically forwarded "
                            + "and will be completed if there are sufficient resources");
        }
```

Method: sendRequest(HttpHeaders, Request)

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	19	[0..11)
NOPM		Number Of Parameters	2	[0..3)
LAA	Lanza-Mari...	Locality Of Attribute Accesses	0,0	[0,33..1,0]
FDP	Lanza-Mari...	Foreign Data Providers	1	[0..3)
NOAV	Lanza-Mari...	Number Of Accessed Variables	4	[0..3)
MND	Lanza-Mari...	Maximum Nesting Depth	1	[0..3)
CINT	Lanza-Mari...	Coupling Intensity	7	[0..7)
CDISP	Lanza-Mari...	Coupling Dispersion	0,75	[0,0..0,5]
HVL	Halstead M...	Halstead Volume	130,028	
HD	Halstead M...	Halstead Difficulty	7,5833	
HL	Halstead M...	Halstead Length	28	
HEF	Halstead M...	Halstead Effort	986,0455	
HVC	Halstead M...	Halstead Vocabulary	25	
HER	Halstead M...	Halstead Errors	0,033	
MMI	Maintainabi...	Maintainability Index	57,2102	[0,0..19,0]

Of course we can't just redelegate smelly code to other methods, so we need to improve this extracted method as well.

The method, called getResponseEntity, had the following metrics:

Method: getResponseEntity(LocalDate, String, Request)

Metric	Metrics Set	Description	Value	Regular Ra...
CND		Condition Nesting Depth	2	[0..2)
LND		Loop Nesting Depth	0	[0..2)
CC		McCabe Cyclomatic Complexity	8	[0..3)
NOL		Number Of Loops	0	
LOC		Lines Of Code	50	[0..11)
NOPM		Number Of Parameters	3	[0..3)
LAA	Lanza-Mari...	Locality Of Attribute Accesses	0,6	[0,33..1,0]
FDP	Lanza-Mari...	Foreign Data Providers	1	[0..3)
NOAV	Lanza-Mari...	Number Of Accessed Variables	10	[0..3)
MND	Lanza-Mari...	Maximum Nesting Depth	2	[0..3)
CINT	Lanza-Mari...	Coupling Intensity	13	[0..7)
CDISP	Lanza-Mari...	Coupling Dispersion	0,5333	[0,0..0,5]
HVL	Halstead M...	Halstead Volume	538,7951	
HD	Halstead M...	Halstead Difficulty	23,2955	
HL	Halstead M...	Halstead Length	97	
HEF	Halstead M...	Halstead Effort	12551,4772	
HVC	Halstead M...	Halstead Vocabulary	47	
HER	Halstead M...	Halstead Errors	0,18	
MMI	Maintainabi...	Maintainability Index	43,5381	[0,0..19,0]

We can improve this also method by extracting the following piece of code:

```
private void setSaveAndPublishRequest(Request request, boolean approved){
    request.setApproved(approved);
    requestRepository.save(request);
    publishRequest();
}
```

This method is called 3 times in the getResponseEntity method, so is a good refactor. Also after refactoring the if else statements, we get the following metrics:

Method: getResponseEntity(String, Request)

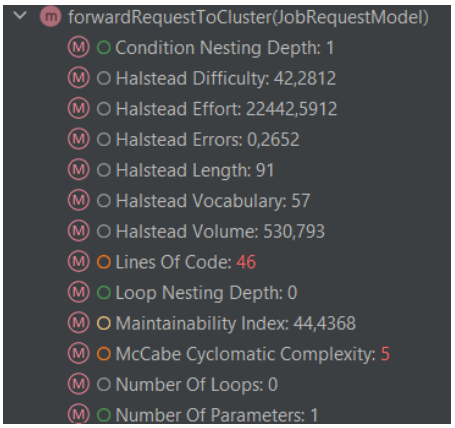
Metric	Metrics Set	Description	Value	Regular Ra...
CND		Condition Nesting Depth	2	[0..2)
LND		Loop Nesting Depth	0	[0..2)
CC		McCabe Cyclomatic Complexity	8	[0..3)
NOL		Number Of Loops	0	
LOC		Lines Of Code	42	[0..11)
NOPM		Number Of Parameters	2	[0..3)
LAA	Lanza-Mari...	Locality Of Attribute Accesses	0,5	[0,33..1,0]
FDP	Lanza-Mari...	Foreign Data Providers	1	[0..3)
NOAV	Lanza-Mari...	Number Of Accessed Variables	6	[0..3)
MND	Lanza-Mari...	Maximum Nesting Depth	2	[0..3)
CINT	Lanza-Mari...	Coupling Intensity	12	[0..7)
CDISP	Lanza-Mari...	Coupling Dispersion	0,5	[0,0..0,5]
HVL	Halstead M...	Halstead Volume	455,3919	
HD	Halstead M...	Halstead Difficulty	21,0833	
HL	Halstead M...	Halstead Length	85	
HEF	Halstead M...	Halstead Effort	9601,1797	
HVC	Halstead M...	Halstead Vocabulary	41	
HER	Halstead M...	Halstead Errors	0,1506	
MMI	Maintainabi...	Maintainability Index	45,6994	[0,0..19,0]

We can now see the finished refactoring. If we compare the original smelly method with this method, we can see the CND has gone down from 4 to 2, which is what we wanted. The CC has gone down from 10 to 8, which is outside the acceptable range, but not further improbable because there need to be a certain amount of if else statements. The LOC has gone down from 59 to 42, which is also a more acceptable range. Ideally this should be lower, but in development there is a compromise between effort and reward, and more refactoring in this method is not worth it.
Lastly, 2 more metrics on which we did not focus, also went down considerably. This shows that refactoring can have more benefits for other metrics than expected. The NOAV has gone down from 15 to 6, which is also now in a more acceptable range. Lastly we can check the CINT, which has gone down from 22 to 12, which is also a good reduction.

2. Method Signature: forwardRequestToCluster(@RequestBody JobRequestModel jobModel)

Location: ScheduleController class (cluster microservice)
Code smell: Cyclomatic Complexity(CC)

Metrics before refactoring



The method-level metrics that need to be improved

- Cyclomatic Complexity(CC) - very high severity
- Lines of Code (LOC) - very high severity

Why those metrics and this method

- This method, as stated before, can be found in the ScheduleController class. This class happens to have one of the highest “Weighted methods per class”(WMC) values in our system. The WMC is influenced by both the CC and the LOC, which happen to be very high in this method. Therefore, we decided to tackle these two here.

Method before refactoring

```java
public ResponseEntity<String> forwardRequestToCluster(@RequestBody JobRequestModel jobModel) {
    // extract job from request model
    Job job = new JobBuilder().requestedThroughFaculty(jobModel.getFacultyId())
            .requestedByUserWithNetId(jobModel.getUserNetId())
            .havingName(jobModel.getJobName())
            .withDescription(jobModel.getJobDescription())
            .needingCpuResources(jobModel.getRequiredCpu())
            .needingGpuResources(jobModel.getRequiredGpu())
            .needingMemoryResources(jobModel.getRequiredMemory())
            .preferredCompletedBeforeDate(jobModel.getPreferredCompletionDate())
            .constructJobInstance();

    // preferred completion date is in the future
    if (job.getPreferredCompletionDate().isBefore(this.dateProvider.getTomorrow())) {
        return ResponseEntity.badRequest()
                .body("The requested job cannot require the cluster to compute it before "
                        + this.dateProvider.getTomorrow() + ".");
    }

    // the resources requested are cpu >= gpu and cpu >= memory
    if (job.getRequiredCpu() < job.getRequiredGpu()
            || job.getRequiredCpu() < job.getRequiredMemory()) {
        return ResponseEntity.badRequest()
                .body("The requested job cannot require more GPU or memory than CPU.");
    }

    // can job ever be scheduled
    if (!this.scheduling.checkIfJobCanBeScheduled(job)) {
        return ResponseEntity.badRequest()
                .body("The requested job requires more resources than are assigned to the "
```

```java
// can job ever be scheduled
if (!this.scheduling.checkIfJobCanBeScheduled(job)) {
    return ResponseEntity.badRequest()
            .body("The requested job requires more resources than are assigned to the "
                    + job.getFacultyId() + " faculty.");
}

// schedule job
var scheduledFor :LocalDate = this.scheduling.scheduleJob(job);
applicationEventPublisher.publishEvent(
        new NotificationEvent( source: this, scheduledFor.toString(), type: "JOB",
                state: "SCHEDULED", message: "Your job has been scheduled by the cluster!", job.getUserNetId()
        ));

// return
return ResponseEntity.ok("Successfully scheduled job.");
}
```

Idea: we will use the “Extract Method Refactoring” strategy on the second if statement, and create a method “areResourcesNodeValid()” in the “Job” class that will compute and return whether the amount of resources is enough or not (second if statement in the first image above).

Code after refactoring

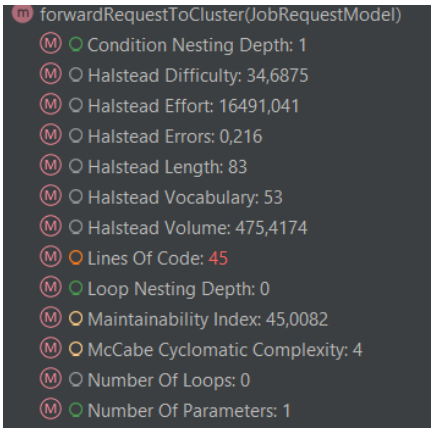
```java
@PostMapping("/request")
public ResponseEntity<String> forwardRequestToCluster(@RequestBody JobRequestModel jobModel) {
    // extract job from request model
    Job job = new JobBuilder().requestedThroughFaculty(jobModel.getFacultyId())
            .requestedByUserWithNetId(jobModel.getUserNetId())
            .havingName(jobModel.getJobName())
            .withDescription(jobModel.getJobDescription())
            .needingCpuResources(jobModel.getRequiredCpu())
            .needingGpuResources(jobModel.getRequiredGpu())
            .needingMemoryResources(jobModel.getRequiredMemory())
            .preferredCompletedBeforeDate(jobModel.getPreferredCompletionDate())
            .constructJobInstance();

    // preferred completion date is in the future
    if (job.getPreferredCompletionDate().isBefore(this.dateProvider.getTomorrow())) {
        return ResponseEntity.badRequest()
                .body("The requested job cannot require the cluster to compute it before "
                        + this.dateProvider.getTomorrow() + ".");
    }

    // the resources requested are cpu >= gpu and cpu >= memory
    if (!job.areResourcesNeededValid()) {
        return ResponseEntity.badRequest()
                .body("The requested job cannot require more GPU or memory than CPU.");
    }

    // can job ever be scheduled
    if (!this.scheduling.checkIfJobCanBeScheduled(job)) {
        return ResponseEntity.badRequest()
                .body("The requested job requires more resources than are assigne
```

Manage multiple Spring B
in the Services tool windo

```java
/**
 * This method calculates whether the amount of resources needed to execute the
 * job are valid. Namely, if the amount gpu and memory required is at most the
 * same as the amount of cpu required.
 *
 * @return true if valid, false if invalid.
 */
public boolean areResourcesNeededValid() {
    if (this.requiredCpu < this.requiredGpu
        || this.requiredCpu < this.requiredMemory) {
        return false;
    }
    return true;
}
```

- After doing this, we expect the cyclomatic complexity to decrease, given that we reduce the number of conditions by one - from five to four. Consequently, this drops the number of possible paths by one as well, which is how we calculate the CC (number of possible paths that a method can take).

Metrics after refactoring

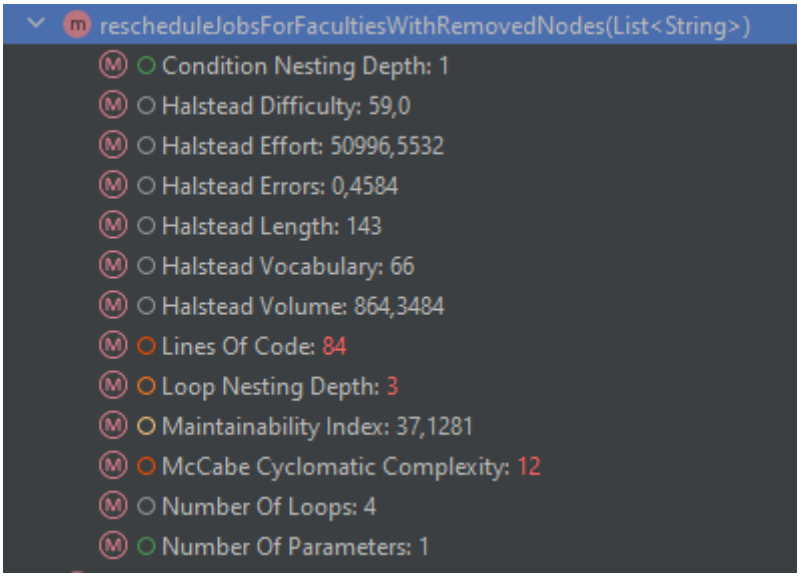


Refactoring evaluation

- As we can see in the image above, our metric points out exactly what we expected to happen.
- Ideally, the metricsTree tool tells us that we should decrease the complexity even more. However, that is not possible since we need those three if statements in order to output different messages for each case, which leads to the CC value of 4. In any case, although we only decreased the complexity by one point, this was enough to bring it down from the “very high” to the “high” category. Furthermore, this reduces the WMC of the class by one as well.
- Aside from the cyclomatic complexity, we also wanted to reduce the LOC. However, this is a trade-off. In order to decrease the LOC of this method, we would have to extract some of its functionality to another method inside that same class. By doing this, we would increase the overall CC of the class, since we cannot further decrease the CC of the method. Given that our main goal in the end is to decrease the CC of the ScheduleController class, we opt to leave the method with a higher LOC, in exchange for a smaller CC.

3. Method signature: rescheduleJobsForFacultiesWithRemovedNodes

Metrics before refactoring:



The method-level metrics that need to be improved:

- Lines Of Code (extreme severity)
- McCable Cyclomatic Complexity (extreme severity)
- Loop nesting Depth

Why those metrics and this method

- RescheduleJobsForFacultiesWithRemovedNodes has a couple extreme severties from the metrics that is an issue that needs to be resolved.
- The amount of lines of code is a serious problem because it makes it harder to understand the method and what it is doing.
- The McCabe Cyclomatic Complexity is also a good indicator that a method is hard to read and to maintain because of all the paths it could go.

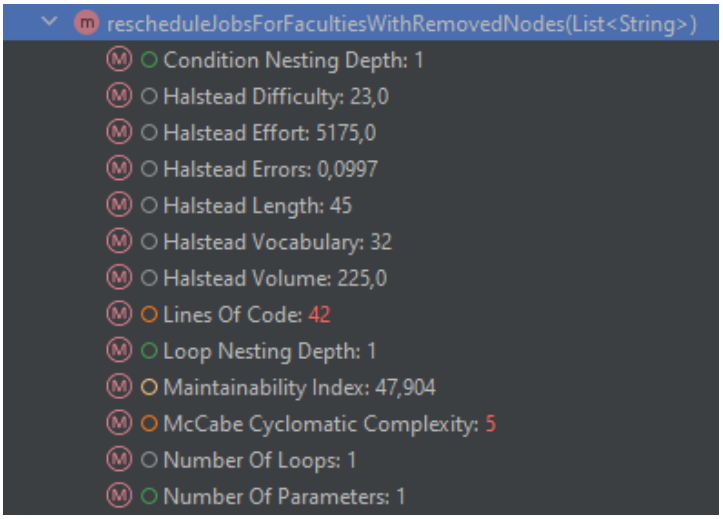
Idea: have helper methods for some of the more complex parts of the method. Hereby it reduces the lines of code of the method and the cyclomatic complexity. The reader of the code does not need to understand what is going on in the helper method to understand what is going on in the original method. That helps with readability and maintainability of the code.

Action plan:

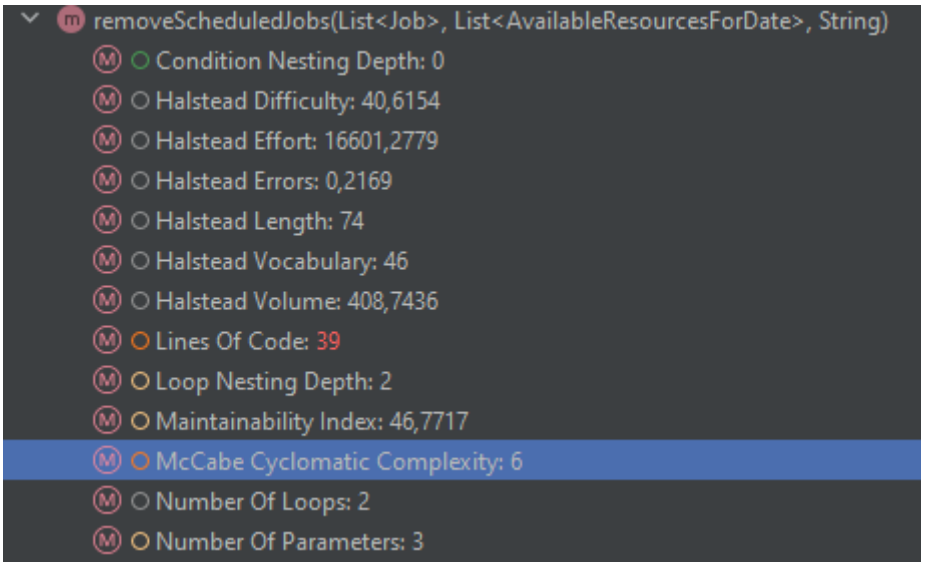
- Identify bits of code that could be helper methods so that it reduces the cyclomatic complexity.
- Create new methods that implement the bits of code that we want to extract.
- Place references to the new methods in the original method
- Check with the test if the functionality still works

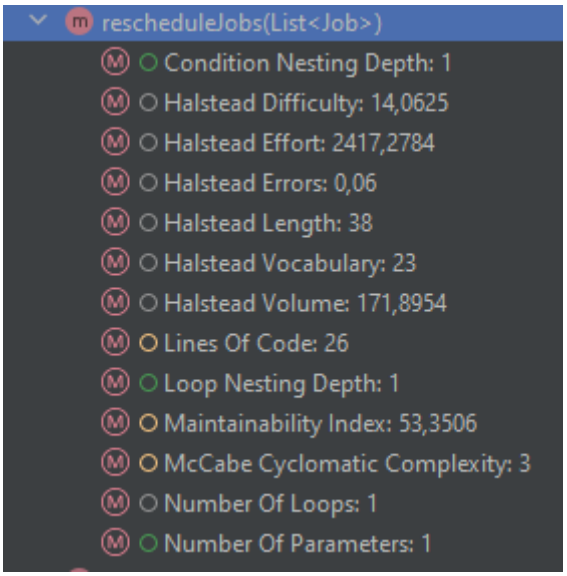
- Write javaDoc for the new methods so it is easy to understand and you don't need to read the code to see what the new method does

Metric after refactoring:



Metrics of the new methods:





Refactoring evaluation: The code is now more spread between methods and the complexity is also spread so it is easier to read. The new method removeScheduledJobs could even be more separated to have even viewer Cyclomatic Complexity but then the code starts getting really shattered which makes it actually harder to read and maintain.

4. Methodname: sendApprovals (HttpHeaders, ApprovalInformation)

Location: Request microservice -> controllers -> JobRequestController

Metric before refactoring

sendApprovals(HttpHeaders, ApprovalInformation)
- Condition Nesting Depth: 1
- Halstead Difficulty: 24.6176
- Halstead Effort: 8063.9016
- Halstead Errors: 0.134
- Halstead Length: 60
- Halstead Vocabulary: 44
- Halstead Volume: 327.5659
- Lines Of Code: 40
- Loop Nesting Depth: 1
- Maintainability Index: 47.2594
- McCabe Cyclomatic Complexity: 4
- Number Of Loops: 2
- Number Of Parameters: 2

The method-level metrics that need to be improved

- Lines of Code (LOC)

Action plan:

- Extract some of the functionality into a separate method => decreased LOC
- Improve CC and readability

Metrics after refactoring:

handleRequests(List<Request>, String)
- Condition Nesting Depth: 1
- Halstead Difficulty: 13.75
- Halstead Effort: 1517.4706
- Halstead Errors: 0.044
- Halstead Length: 27
- Halstead Vocabulary: 17
- Halstead Volume: 110.3615
- Lines Of Code: 16
- Loop Nesting Depth: 1
- Maintainability Index: 59.2531
- McCabe Cyclomatic Complexity: 4
- Number Of Loops: 2
- Number Of Parameters: 2

sendApprovals(HttpHeaders, ApprovalInformation)
- Condition Nesting Depth: 0
- Halstead Difficulty: 12.0
- Halstead Effort: 2400.5045
- Halstead Errors: 0.0598
- Halstead Length: 39
- Halstead Vocabulary: 35
- Halstead Volume: 200.042
- Lines Of Code: 27
- Loop Nesting Depth: 0
- Maintainability Index: 52.6644
- McCabe Cyclomatic Complexity: 1
- Number Of Loops: 0
- Number Of Parameters: 2

A lot of the metrics are now improved, and most importantly the LOC is revised significantly. The refactoring also made the code much more readable with two methods, each with separated concerns.
The main operation of the refactoring is breaking up code into two methods each one with a clearly defined purpose and function. This operation significantly improved metrics such as Halstead effort, Halstead errors which signify the overall difficulty of maintaining the method.

5. Methodname: getReservedResource

Location: Request microservice -> services-> RequestAllocationService

Metrics before refactoring:

```
getReservedResource(String, LocalDate, String)
    Condition Nesting Depth: 0
    Halstead Difficulty: 16.4474
    Halstead Effort: 5208.0104
    Halstead Errors: 0.1002
    Halstead Length: 58
    Halstead Vocabulary: 44
    Halstead Volume: 316.647
    Lines Of Code: 29
    Loop Nesting Depth: 0
    Maintainability Index: 50.5032
    McCabe Cyclomatic Complexity: 2
    Number Of Loops: 0
    Number Of Parameters: 3
```

The method-level metrics that need to be improved

- Halstead Metrics

Action plan:

- Clear redundant variables
- Simplify business logic within class
- Remove overall bulk in method

Metrics after refactoring

```
getReservedResource(String, LocalDate, String)
    Condition Nesting Depth: 0
    Halstead Difficulty: 16.6667
    Halstead Effort: 4974.076
    Halstead Errors: 0.0971
    Halstead Length: 55
    Halstead Vocabulary: 43
    Halstead Volume: 298.4446
    Lines Of Code: 27
    Loop Nesting Depth: 0
    Maintainability Index: 51.3586
    McCabe Cyclomatic Complexity: 2
    Number Of Loops: 0
    Number Of Parameters: 3
```

This method is essential for the functionality of the service class. I removed the unnecessary variable availableResources, as it is not used in the method. Instead of using

Objects.requireNonNull(availableResources).getResourceList(), I returned the listOfResources directly, since it is the same thing. I replaced the System.out.println statement with System.err.println to indicate that this is an error message. Also added a message for the error so that it will be clear why the error is happening. Returned an empty list in the case of an exception to make sure that the method always returns a list. These actions significantly improved the Halstead metrics which shows the method now is less bulky and hard-to-read.

Class level code smells

1. Class name: ScheduleController

Metrics before refactoring

Class: ScheduleController

Metric	Metrics Set	Description	Value	Regular Ra...
CHVL	Halstead M...	Halstead Volume	3068,0326	
CHD	Halstead M...	Halstead Difficulty	121,875	
CHL	Halstead M...	Halstead Length	439	
CHEF	Halstead M...	Halstead Effort	373916,4704	
CHVC	Halstead M...	Halstead Vocabulary	127	
CHER	Halstead M...	Halstead Errors	1,7301	
WMC	Chidamber-...	Weighted Methods Per Class	39	[0..12)
DIT	Chidamber-...	Depth Of Inheritance Tree	1	[0..3)
RFC	Chidamber-...	Response For A Class	64	[0..45)
LCOM	Chidamber-...	Lack Of Cohesion Of Methods	1	
NOC	Chidamber-...	Number Of Children	0	[0..2)
NOA	Lorenz-Kid...	Number Of Attributes	6	[0..4)
NOO	Lorenz-Kid...	Number Of Operations	19	
NOOM	Lorenz-Kid...	Number Of Overridden Methods	0	[0..3)
NOAM	Lorenz-Kid...	Number Of Added Methods	6	
SIZE2	Li-Henry M...	Number Of Attributes And Methods	25	
NOM	Li-Henry M...	Number Of Methods	7	[0..7)
MPC	Li-Henry M...	Message Passing Coupling	108	
DAC	Li-Henry M...	Data Abstraction Coupling	6	
NCSS	Chr. Clemen...	Non-Commenting Source Statement..	87	[0..1000)
CMI	Maintainabi...	Maintainability Index	23,7398	[0,0..19,0]

The class-level metrics that need to be improved

- Weighted Methods per Class(WMC) - very high

Why that metric and this Class

- This class has one of the highest WMC in our system, and therefore we considered it to be a priority.
- The WMC is influenced by both the cyclomatic complexity(CC) and the line of codes (LOC), which are very important metrics. Thus, in order to reduce the WMC, we need to reduce the CC and/or the LOC, which leads us to improve the overall implementation of not only the class, but its methods.

Idea

- First, we will improve some method-level metrics (i.e. reduce the CC of the method forwardRequestToCluster), which will directly impact the WMC of the class.
- After that, we will split the ScheduleController class in two; One will remain as the ScheduleController, which will handle endpoints related to the schedule, while the other will be named FacultyResourcesInformationController, which will handle the endpoints related to faculties' resources.

Action plan

- Our first action is to decrease the cyclomatic complexity of the method "forwardRequestToCluster" in this microservice. In order to see how we did that, refer back to the method-level code smells section.
- After this first action, our metric improved by one:

Class: ScheduleController

Metric	Metrics Set	Description	Value	Regular Ra...
CHVL	Halstead M...	Halstead Volume	2997,2586	
CHD	Halstead M...	Halstead Difficulty	114,9231	
CHL	Halstead M...	Halstead Length	431	
CHEF	Halstead M...	Halstead Effort	344454,1818	
CHVC	Halstead M...	Halstead Vocabulary	124	
CHER	Halstead M...	Halstead Errors	1,6379	
WMC	Chidamber-...	Weighted Methods Per Class	38	[0..12)
DIT	Chidamber-...	Depth Of Inheritance Tree	1	[0..3)
RFC	Chidamber-...	Response For A Class	62	[0..45)
LCOM	Chidamber-...	Lack Of Cohesion Of Methods	1	
NOC	Chidamber-...	Number Of Children	0	[0..2)
NOA	Lorenz-Kid...	Number Of Attributes	6	[0..4)
NOO	Lorenz-Kid...	Number Of Operations	19	
NOOM	Lorenz-Kid...	Number Of Overridden Methods	0	[0..3)
NOAM	Lorenz-Kid...	Number Of Added Methods	6	
SIZE2	Li-Henry M...	Number Of Attributes And Methods	25	
NOM	Li-Henry M...	Number Of Methods	7	[0..7)
MPC	Li-Henry M...	Message Passing Coupling	105	
DAC	Li-Henry M...	Data Abstraction Coupling	6	
NCSS	Chr. Clemen...	Non-Commenting Source Statement..	87	[0..1000)
CMI	Maintainabi...	Maintainability Index	23,8565	[0,0..19,0]

- Next, we will split the controller and controller test classes into two controllers and two test classes, as explained before.

Metrics after both refactoring actions

Class: ScheduleController

Metric	Metrics Set	Description	Value	Regular Ra.
CHVL	Halstead M...	Halstead Volume	713,2385	
CHD	Halstead M...	Halstead Difficulty	36,75	
CHL	Halstead M...	Halstead Length	118	
CHEF	Halstead M...	Halstead Effort	26211,5151	
CHVC	Halstead M...	Halstead Vocabulary	66	
CHER	Halstead M...	Halstead Errors	0,2941	
WMC	Chidamber-...	Weighted Methods Per Class	6	[0..12)
DIT	Chidamber-...	Depth Of Inheritance Tree	1	[0..3)
RFC	Chidamber-...	Response For A Class	35	[0..45)
LCOM	Chidamber-...	Lack Of Cohesion Of Methods	2	
NOC	Chidamber-...	Number Of Children	0	[0..2)
NOA	Lorenz-Kid...	Number Of Attributes	4	[0..4)
NOO	Lorenz-Kid...	Number Of Operations	15	
NOOM	Lorenz-Kid...	Number Of Overridden Methods	0	[0..3)
NOAM	Lorenz-Kid...	Number Of Added Methods	2	
SIZE2	Li-Henry M...	Number Of Attributes And Methods	19	
NOM	Li-Henry M...	Number Of Methods	3	[0..7)
MPC	Li-Henry M...	Message Passing Coupling	36	
DAC	Li-Henry M...	Data Abstraction Coupling	4	
NCSS	Chr. Clemen...	Non-Commenting Source Statement..	18	[0..1000)
CMI	Maintainabi...	Maintainability Index	39,9477	[0,0..19,0]

Class: FacultyResourcesInformationController

Metric	Metrics Set	Description	Value	Regular Ra
CHVL	Halstead M...	Halstead Volume	2161,8889	
CHD	Halstead M...	Halstead Difficulty	79,0811	
CHL	Halstead M...	Halstead Length	341	
CHEF	Halstead M...	Halstead Effort	170964,5075	
CHVC	Halstead M...	Halstead Vocabulary	81	
CHER	Halstead M...	Halstead Errors	1,0268	
WMC	Chidamber-...	Weighted Methods Per Class	33	[0..12)
DIT	Chidamber-...	Depth Of Inheritance Tree	1	[0..3)
RFC	Chidamber-...	Response For A Class	32	[0..45)
LCOM	Chidamber-...	Lack Of Cohesion Of Methods	1	
NOC	Chidamber-...	Number Of Children	0	[0..2)
NOA	Lorenz-Kid...	Number Of Attributes	6	[0..4)
NOO	Lorenz-Kid...	Number Of Operations	17	
NOOM	Lorenz-Kid...	Number Of Overridden Methods	0	[0..3)
NOAM	Lorenz-Kid...	Number Of Added Methods	4	
SIZE2	Li-Henry M...	Number Of Attributes And Methods	23	
NOM	Li-Henry M...	Number Of Methods	5	[0..7)
MPC	Li-Henry M...	Message Passing Coupling	69	
DAC	Li-Henry M...	Data Abstraction Coupling	6	
NCSS	Chr. Clemen...	Non-Commenting Source Statement..	73	[0..1000)
CMI	Maintainabi...	Maintainability Index	27,4147	[0,0..19,0]

Refactoring Evaluation: Both new classes now have significantly fewer WMC values now. As explained before, this is highly related to LOC and CC, which means that we reduced both metrics. This can be confirmed by our explanation and by the screenshots shown above. The splitting of the controller class in two makes sense, because now all the endpoints relate to the others in the controller. Namely, the endpoints in the ScheduleController all relate to tasks that somehow access or change the schedule, while the endpoints in the FacultyResourcesInformationController to tasks that relate to the faculties' resources. Hence, by doing this splitting we guarantee that each controller is responsible for one part of the cluster.

2. ClassName: DataProcessingService

Metrics before refactoring:

Class: DataProcessingService

Metric	Metrics Set	Description	Value	Regular Ra...
CHVL	Halstead M...	Halstead Volume	2660,0746	
CHD	Halstead M...	Halstead Difficulty	134,2258	
CHL	Halstead M...	Halstead Length	397	
CHEF	Halstead M...	Halstead Effort	357050,6541	
CHVC	Halstead M...	Halstead Vocabulary	104	
CHER	Halstead M...	Halstead Errors	1,6776	
WMC	Chidamber-...	Weighted Methods Per Class	38	[0..12)
DIT	Chidamber-...	Depth Of Inheritance Tree	1	[0..3)
RFC	Chidamber-...	Response For A Class	77	[0..45)
LCOM	Chidamber-...	Lack Of Cohesion Of Methods	1	
NOC	Chidamber-...	Number Of Children	0	[0..2)
NOA	Lorenz-Kid...	Number Of Attributes	3	[0..4)
NOO	Lorenz-Kid...	Number Of Operations	42	
NOOM	Lorenz-Kid...	Number Of Overridden Methods	0	[0..3)
NOAM	Lorenz-Kid...	Number Of Added Methods	28	
SIZE2	Li-Henry M...	Number Of Attributes And Methods	44	
NOM	Li-Henry M...	Number Of Methods	29	[0..7)
MPC	Li-Henry M...	Message Passing Coupling	134	
DAC	Li-Henry M...	Data Abstraction Coupling	3	
NCSS	Chr. Clemen...	Non-Commenting Source Statements	66	[0..1000)
CMI	Maintainabi...	Maintainability Index	21,9798	[0,0..19,0]

The class-level metrics that need to be improved:

- Number of Methods (extreme severity)

- Weighted Methods per Class (very high severity)
- Response For a Class (very high severity)

Why those metrics and this class:

- DataProcessingService is the only class to have 3 metrics that are extremely off norm. Therefore, improving it should be a priority.
- The DataProcessingService is a very large class, one of the largest in the system. The number of methods and interactions that those methods handle is thus much larger than average and can make this class both difficult to understand and test.
- Reducing the number of methods and method calls should greatly improve readability and maintainability of the class and facilitate potential future changes.

Idea: split the class into a node-oriented processing/access service and a schedule-oriented one. The NOM (and thus WMC) will naturally decrease as a result of each new class now containing fewer methods. As there will be fewer methods, there will also be fewer method calls, so RFC should also go down substantially. To achieve best results, the split should keep around half of DataProcessingService's methods in each of the new classes.

Action plan:

- identify all methods that can be removed from the main DataProcessingService class. The number of such methods should be as close to half the number of methods in the old class as possible without completely breaking the project's structure
- create a new class for the outlined methods and extract them
- change the test class to account for the methods now being in the new class (NodeDataProcessingService), add new test class and move relevant tests there
- check for any references to the now-moved methods of DataProcessingService in other classes and reorient them towards NodeDataProcessingService
- if necessary, change parameters/fields from DataProcessingService to NodeDataProcessingService, or add new ones
- rename DataProcessingService to SchedulingDataProcessingService (as this is the main focus of this downsized service) and rename the test
- likewise, change the names of all parameters/fields that are of that class

Metrics of the two classes after refactoring:

Class: NodeDataProcessingService

Metric	Metrics Set	Description	Value	Regular Ra...
CHVL	Halstead M...	Halstead Volume	346,7937	
CHD	Halstead M...	Halstead Difficulty	35,4167	
CHL	Halstead M...	Halstead Length	70	
CHEF	Halstead M...	Halstead Effort	12282,2784	
CHVC	Halstead M...	Halstead Vocabulary	31	
CHER	Halstead M...	Halstead Errors	0,1774	
WMC	Chidamber-...	Weighted Methods Per Class	10	[0..12)
DIT	Chidamber-...	Depth Of Inheritance Tree	1	[0..3)
RFC	Chidamber-...	Response For A Class	24	[0..45)
LCOM	Chidamber-...	Lack Of Cohesion Of Methods	1	
NOC	Chidamber-...	Number Of Children	0	[0..2)
NOA	Lorenz-Kid...	Number Of Attributes	1	[0..4)
NOO	Lorenz-Kid...	Number Of Operations	23	
NOOM	Lorenz-Kid...	Number Of Overridden Methods	0	[0..3)
NOAM	Lorenz-Kid...	Number Of Added Methods	9	
SIZE2	Li-Henry M...	Number Of Attributes And Methods	23	
NOM	Li-Henry M...	Number Of Methods	10	[0..7)
MPC	Li-Henry M...	Message Passing Coupling	15	
DAC	Li-Henry M...	Data Abstraction Coupling	1	
NCSS	Chr. Clemen...	Non-Commenting Source Statements	10	[0..1000)
CMI	Maintainabi...	Maintainability Index	47,9625	[0,0..19,0]

Class: DataProcessingService

Metric	Metrics Set	Description	Value	Regular Ra...
CHVL	Halstead M...	Halstead Volume	2214,0413	
CHD	Halstead M...	Halstead Difficulty	112,6552	
CHL	Halstead M...	Halstead Length	337	
CHEF	Halstead M...	Halstead Effort	249423,2089	
CHVC	Halstead M...	Halstead Vocabulary	95	
CHER	Halstead M...	Halstead Errors	1,3208	
WMC	Chidamber-...	Weighted Methods Per Class	29	[0..12)
DIT	Chidamber-...	Depth Of Inheritance Tree	1	[0..3)
RFC	Chidamber-...	Response For A Class	63	[0..45)
LCOM	Chidamber-...	Lack Of Cohesion Of Methods	1	
NOC	Chidamber-...	Number Of Children	0	[0..2)
NOA	Lorenz-Kid...	Number Of Attributes	3	[0..4)
NOO	Lorenz-Kid...	Number Of Operations	33	
NOOM	Lorenz-Kid...	Number Of Overridden Methods	0	[0..3)
NOAM	Lorenz-Kid...	Number Of Added Methods	19	
SIZE2	Li-Henry M...	Number Of Attributes And Methods	35	
NOM	Li-Henry M...	Number Of Methods	20	[0..7)
MPC	Li-Henry M...	Message Passing Coupling	119	
DAC	Li-Henry M...	Data Abstraction Coupling	3	
NCSS	Chr. Clemen...	Non-Commenting Source Statements	57	[0..1000)
CMI	Maintainabi...	Maintainability Index	23,5171	[0,0..19,0]

Refactoring evaluation: both new classes now have significantly fewer methods. The WMC and RFC metrics have been considerably improved as well, as a natural consequence of the classes being smaller and more concise. While NOM for SchedulingDataProcessingService remains fairly high, we deemed it to be at an acceptable level, considering that we would otherwise have to significantly alter the structure of the microservice to lower it further. Moreover, all the methods that are left in that class relate to similar concepts and tasks, so readability and maintainability should nevertheless be much better.

3. Class name: NodeController.java

Metrics before refactoring:

Class: NodeController

Metric	Metrics Set	Description	Value	Regular Ra.
CHVL	Halstead M...	Halstead Volume	2339,3214	
CHD	Halstead M...	Halstead Difficulty	73,0189	
CHL	Halstead M...	Halstead Length	343	
CHEF	Halstead M...	Halstead Effort	170814,5992	
CHVC	Halstead M...	Halstead Vocabulary	113	
CHER	Halstead M...	Halstead Errors	1,0262	
WMC	Chidamber-...	Weighted Methods Per Class	25	[0..12)
DIT	Chidamber-...	Depth Of Inheritance Tree	1	[0..3)
RFC	Chidamber-...	Response For A Class	50	[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	19	
NOOM	Lorenz-Kid...	Number Of Overridden Methods	0	[0..3)
NOAM	Lorenz-Kid...	Number Of Added Methods	5	
SIZE2	Li-Henry M...	Number Of Attributes And Methods	22	
NOM	Li-Henry M...	Number Of Methods	6	[0..7)
MPC	Li-Henry M...	Message Passing Coupling	112	
DAC	Li-Henry M...	Data Abstraction Coupling	4	
NCSS	Chr. Clemen...	Non-Commenting Source Statements	79	[0..1000)
CMI	Maintainabi...	Maintainability Index	26,8865	[0,0..19,0]

The class-level metrics that need to be improved:

- Weighted Methods per Class (high severity)
- Response For a Class (high severity)
- Number of Attributes (high severity)

Why those metrics and this class:

- NodeController is an important and integral part of the client's requirements, and thus this class has a higher priority to make sure all the class metrics are in acceptable ranges.
- All of the metrics that need to be improved are outside the recommended regular range, so when all of those metrics mentioned above are back in their regular ranges, we can say with high certainty that this class is well coded, maintainable and readable for future developers.

Idea: the plan is to split up the class into 2 new classes. This refactoring should be good to make sure all the mentioned metrics get to their regular values. One class will be called NodeAccessController, which will contain all endpoints that are related to accessing and updating an existing node. The second class, which will be called NodeCreationAndDeletionController, which only handles the creation or deletion of a node.

Action plan:

- Split up all methods from the original class into the 2 new classes, so that the method's job fits into the theme of the new classes correctly.
- Create the two classes, move the existing methods accordingly, and delete the old class.
- Because the controller classes' methods are not called by other code in the Cluster Microservice, we will not need to alter code in any other files in the source code.

- create a new class for the outlined methods and extract them
- change the test class to account for the methods now being in the new class (NodeDataProcessingService), add new test class and move relevant tests there
- check for any references to the now-moved methods of DataProcessingService in other classes and reorient them towards NodeDataProcessingService
- if necessary, change parameters/fields from DataProcessingService to NodeDataProcessingService, or add new ones
- rename DataProcessingService to SchedulingDataProcessingService (as this is the main focus of this downsized service) and rename the test
- likewise, change the names of all parameters/fields that are of that class

Metrics of the two classes after refactoring:

Class: NodeCreationAndDeletionController

Metric	Metrics Set	Description	Value	Regular Ra
CHVL	Halstead M...	Halstead Volume	1756,7795	
CHD	Halstead M...	Halstead Difficulty	60,625	
CHL	Halstead M...	Halstead Length	265	
CHEF	Halstead M...	Halstead Effort	106504,7574	
CHVC	Halstead M...	Halstead Vocabulary	99	
CHER	Halstead M...	Halstead Errors	0,749	
WMC	Chidamber-...	Weighted Methods Per Class	18	[0..12)
DIT	Chidamber-...	Depth Of Inheritance Tree	1	[0..3)
RFC	Chidamber-...	Response For A Class	46	[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	17	
NOOM	Lorenz-Kid...	Number Of Overridden Methods	0	[0..3)
NOAM	Lorenz-Kid...	Number Of Added Methods	3	
SIZE2	Li-Henry M...	Number Of Attributes And Methods	20	
NOM	Li-Henry M...	Number Of Methods	4	[0..7)
MPC	Li-Henry M...	Message Passing Coupling	87	
DAC	Li-Henry M...	Data Abstraction Coupling	4	
NCSS	Chr. Clemen...	Non-Commenting Source Statements	60	[0..1000)
CMI	Maintainabi...	Maintainability Index	31,3039	[0,0..19,0]

Class: NodeAccessController

Metric	Metrics Set	Description	Value	Regular Ra...
CHVL	Halstead M...	Halstead Volume	544,7922	
CHD	Halstead M...	Halstead Difficulty	29,4545	
CHL	Halstead M...	Halstead Length	93	
CHEF	Halstead M...	Halstead Effort	16046,6076	
CHVC	Halstead M...	Halstead Vocabulary	58	
CHER	Halstead M...	Halstead Errors	0,2121	
WMC	Chidamber-...	Weighted Methods Per Class	8	[0..12)
DIT	Chidamber-...	Depth Of Inheritance Tree	1	[0..3)
RFC	Chidamber-...	Response For A Class	25	[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	16	
NOOM	Lorenz-Kid...	Number Of Overridden Methods	0	[0..3)
NOAM	Lorenz-Kid...	Number Of Added Methods	2	
SIZE2	Li-Henry M...	Number Of Attributes And Methods	17	
NOM	Li-Henry M...	Number Of Methods	3	[0..7)
MPC	Li-Henry M...	Message Passing Coupling	25	
DAC	Li-Henry M...	Data Abstraction Coupling	2	
NCSS	Chr. Clemen...	Non-Commenting Source Statements	21	[0..1000)
CMI	Maintainabi...	Maintainability Index	41,6205	[0,0..19,0]

Refactoring evaluation: Both new classes are created successfully and the new metrics are better. NodeAccessController now has acceptable metrics in the regular range for the WMC, RFC and NOA. This is a very successful refactoring and no more work should be done in this class.
However in the NodeCreationAndDeletionController, the metrics are not where we wanted them. The WMC, RFC and NOA are all not where we would like them. We can't easily refactor this class further without splitting it up again, but from a logical standpoint, there will be too many classes that handle a node's creation and deletion, which we think should be contained in one file. So we accept that this refactoring has not yielded the ideal results. However, still, the WMC went down from 25 to 18, and the RFC went down from 50 to 46. The NOA stayed the same at 4, but 4 is not further easily improvable, so we accept this too.
In conclusion, although this refactoring did not yield the ideal results, it still improved the code quality in general, and from a design pattern, the 2 new NodeController classes now have clear distinct functionalities, which is good for future development and maintainability.

4. Class name: JobRequestController.java

Metrics before refactoring:

Class: JobRequestController

Metric	Metrics Set	Description	Value	Regular Ra...
CHVL	Halstead M...	Halstead Volume	1594.7047	
CHD	Halstead M...	Halstead Difficulty	58.32	
CHL	Halstead M...	Halstead Length	238	
CHEF	Halstead M...	Halstead Effort	93003.1754	
CHVC	Halstead M...	Halstead Vocabulary	104	
CHER	Halstead M...	Halstead Errors	0.6842	
WMC	Chidamber-...	Weighted Methods Per Class	16	[0..12)
DIT	Chidamber-...	Depth Of Inheritance Tree	1	[0..3)
RFC	Chidamber-...	Response For A Class	39	[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	16	
NOOM	Lorenz-Kid...	Number Of Overridden Methods	0	[0..3)
NOAM	Lorenz-Kid...	Number Of Added Methods	3	
SIZE2	Li-Henry M...	Number Of Attributes And Met...	20	
NOM	Li-Henry M...	Number Of Methods	4	[0..7)
MPC	Li-Henry M...	Message Passing Coupling	74	
DAC	Li-Henry M...	Data Abstraction Coupling	4	
NCSS	Chr. Clemen...	Non-Commenting Source State...	65	[0..1000)
CMI	Maintainabi...	Maintainability Index	30.9451	[0.0..19.0]

The class-level metrics that need to be improved:

- Weighted Methods per Class (high severity)
- Number of Attributes (high severity)
- Maintainability Index (high severity)

Why those metrics and this class:

- The controller class is an integral part of one of our microservices, thus all of the metrics in it have to be in the sufficient range, if we want to make sure that the Request handling microservice is implemented correctly and does its job optimally
- When the aforementioned metrics are brought to the acceptable range, we can conclude that the class is sufficiently maintainable, scalable, and robust.

Idea:

- Split the class into two separate classes with fewer methods and attributes. One of the new classes - SendingRequestsController, will only be responsible for the functionality of sending requests to the cluster with all of the authentication and time-specific functionality. It will also broadcast all pending requests, so a faculty-privileged account can approve or decline them. The other one will be ApprovingRequestsController, which will contain the functionality for sending approvals. Thus, in the final communication with the cluster - authorization of sender and the sending of the approved request, will be in it. Thus, in the future if additional authorization or filters on the approvals are built on this class it can remain robust and maintainable.

Action plan:

- Split up all methods from the JobRequestController into two separate classes based on the new themes for them and divide the methods accordingly.

- The method names and endpoints will remain the same, so that we don't have issues when the other microservices call them.
- Split the test class into two separate ones.
- Maybe the complexity of the sendRequests method can be reduced by splitting it into several methods or using functional programming. That way the SendRequestsController class will not consist of only 2 methods, one of which takes up most of the lines of code.

Metrics of the two classes after refactoring:

Class: SendingRequestsController

Metric	Metrics Set	Description	Value	Regular Ra...
CHVL	Halstead M...	Halstead Volume	1118.6555	
CHD	Halstead M...	Halstead Difficulty	36.2326	
CHL	Halstead M...	Halstead Length	175	
CHEF	Halstead M...	Halstead Effort	40531.7522	
CHVC	Halstead M...	Halstead Vocabulary	84	
CHER	Halstead M...	Halstead Errors	0.3933	
WMC	Chidamber-...	Weighted Methods Per Class	11	[0..12)
DIT	Chidamber-...	Depth Of Inheritance Tree	1	[0..3)
RFC	Chidamber-...	Response For A Class	26	[0..45)
LCOM	Chidamber-...	Lack Of Cohesion Of Methods	1	
NOC	Chidamber-...	Number Of Children	0	[0..2)
NOA	Lorenz-Kid...	Number Of Attributes	5	[0..4)
NOO	Lorenz-Kid...	Number Of Operations	14	
NOOM	Lorenz-Kid...	Number Of Overridden Methods	0	[0..3)
NOAM	Lorenz-Kid...	Number Of Added Methods	1	
SIZE2	Li-Henry M...	Number Of Attributes And Met...	19	
NOM	Li-Henry M...	Number Of Methods	2	[0..7)
MPC	Li-Henry M...	Message Passing Coupling	49	
DAC	Li-Henry M...	Data Abstraction Coupling	5	
NCSS	Chr. Clemen...	Non-Commenting Source State...	49	[0..1000)
CMI	Maintainabi...	Maintainability Index	36.5844	[0.0..19.0]

Class: ApprovingRequestsController

Metric	Metrics Set	Description	Value	Regular Ra...
CHVL	Halstead M...	Halstead Volume	583.5112	
CHD	Halstead M...	Halstead Difficulty	32.5385	
CHL	Halstead M...	Halstead Length	98	
CHEF	Halstead M...	Halstead Effort	18986.558	
CHVC	Halstead M...	Halstead Vocabulary	62	
CHER	Halstead M...	Halstead Errors	0.2372	
WMC	Chidamber-...	Weighted Methods Per Class	7	[0..12)
DIT	Chidamber-...	Depth Of Inheritance Tree	1	[0..3)
RFC	Chidamber-...	Response For A Class	27	[0..45)
LCOM	Chidamber-...	Lack Of Cohesion Of Methods	1	
NOC	Chidamber-...	Number Of Children	0	[0..2)
NOA	Lorenz-Kid...	Number Of Attributes	3	[0..4)
NOO	Lorenz-Kid...	Number Of Operations	16	
NOOM	Lorenz-Kid...	Number Of Overridden Methods	0	[0..3)
NOAM	Lorenz-Kid...	Number Of Added Methods	3	
SIZE2	Li-Henry M...	Number Of Attributes And Met...	19	
NOM	Li-Henry M...	Number Of Methods	4	[0..7)
MPC	Li-Henry M...	Message Passing Coupling	26	
DAC	Li-Henry M...	Data Abstraction Coupling	3	
NCSS	Chr. Clemen...	Non-Commenting Source State...	21	[0..1000)
CMI	Maintainabi...	Maintainability Index	39.0989	[0.0..19.0]

Evaluation of refactoring:

The refactoring was generally a success. The Weighted Methods per Class (WMC) metric was reduced from 16 to 11 and 7 for the two new classes, both within the acceptable range. The number of attributes for the new SendingRequestsController is still a bit high still, but all of those attributes are necessary for the functioning of the methods, so reduction is not possible without changing the logic of the microservice. The maintainability index remained the same, but that would require full refactoring of the methods in it as well. Although the refactoring was not perfect, the increased modularity and scalability of the two separate functionalities of the microservice is a positive outcome.

5. Name Class: JobSchedulingService.java

Metrics before refactoring:

Class: JobSchedulingService

Metric	Metrics Set	Description	Value	Regular Range
CHVL	Halstead M...	Halstead Volume	2144,3168	
CHD	Halstead M...	Halstead Difficulty	84,3617	
CHL	Halstead M...	Halstead Length	315	
CHEF	Halstead M...	Halstead Effort	180898,2152	
CHVC	Halstead M...	Halstead Vocabulary	112	
CHER	Halstead M...	Halstead Errors	1,0662	
WMC	Chidamber...	Weighted Methods Per Class	28	[0..12)
DIT	Chidamber...	Depth Of Inheritance Tree	1	[0..3)
CBO	Chidamber...	Coupling Between Objects	11	[0..14)
RFC	Chidamber...	Response For A Class	55	[0..45)
LCOM	Chidamber...	Lack Of Cohesion Of Methods	1	
NOC	Chidamber...	Number Of Children	0	[0..2)
NOA	Lorenz-Kid...	Number Of Attributes	5	[0..4)
NOO	Lorenz-Kid...	Number Of Operations	23	
NOOM	Lorenz-Kid...	Number Of Overridden Methods	0	[0..3)
NOAM	Lorenz-Kid...	Number Of Added Methods	10	
SIZE2	Li-Henry M...	Number Of Attributes And Methods	28	
NOM	Li-Henry M...	Number Of Methods	11	[0..7)
MPC	Li-Henry M...	Message Passing Coupling	93	
DAC	Li-Henry M...	Data Abstraction Coupling	5	
ATFD	Lanza-Mari...	Access To Foreign Data	2	[0..6)
NOPA	Lanza-Mari...	Number Of Public Attributes	0	[0..3)
NOAC	Lanza-Mari...	Number Of Accessor Methods	5	[0..4)
WOC	Lanza-Mari...	Weight Of A Class	0,5	[0,5..1,0]
TCC	Bieman-Ka...	Tight Class Cohesion	0,3778	[0,33..1,0]
NCSS	Chr. Cleme...	Non-Commenting Source Statements	58	[0..1000)
CMI	Maintainab...	Maintainability Index	26,6181	[0,0..19,0]

The Class-level metrics that need to be improved:

- Number of methods (High severity)
- Weighted Methods per Class(High severity)

Why those metrics and this class:

- Large classes are often harder to keep up and more of a mess to read and find methods in.
- This class had a clear split between 2 groups of methods. That is why we decided to split this class.

Idea: split the class into a JobSchedulingService and a JobReschedulingService. This will decrease the amount of methods that the classes have and the overall weight of the class. This will greatly improve the readability of the class and therefore maintainability.

Action plan:

- identify which methods work for rescheduling the jobs after a node is removed.
- Extracting the methods we identified and altering some of the method calls inside of them
- Split the test classes into the two new classes.
- Test the code to make sure it is still working correctly.

Metrics of the two classes after refactoring:

Class: JobSchedulingService

Metric	Metrics Set	Description	Value	Regular Ra...
CHVL	Halstead M...	Halstead Volume	1064,6498	
CHD	Halstead M...	Halstead Difficulty	47,7931	
CHL	Halstead M...	Halstead Length	172	
CHEF	Halstead M...	Halstead Effort	50882,9192	
CHVC	Halstead M...	Halstead Vocabulary	73	
CHER	Halstead M...	Halstead Errors	0,4577	
WMC	Chidamber...	Weighted Methods Per Class	16	[0..12)
DIT	Chidamber...	Depth Of Inheritance Tree	1	[0..3)
RFC	Chidamber...	Response For A Class	40	[0..45)
LCOM	Chidamber...	Lack Of Cohesion Of Methods	1	
NOC	Chidamber...	Number Of Children	0	[0..2)
NOA	Lorenz-Kid...	Number Of Attributes	5	[0..4)
NOO	Lorenz-Kid...	Number Of Operations	22	
NOOM	Lorenz-Kid...	Number Of Overridden Methods	0	[0..3)
NOAM	Lorenz-Kid...	Number Of Added Methods	9	
SIZE2	Li-Henry M...	Number Of Attributes And Methods	27	
NOM	Li-Henry M...	Number Of Methods	10	[0..7)
MPC	Li-Henry M...	Message Passing Coupling	46	
DAC	Li-Henry M...	Data Abstraction Coupling	5	
NCSS	Chr. Cleme...	Non-Commenting Source Statements	28	[0..1000)
CMI	Maintainab...	Maintainability Index	34,4329	[0,0..19,0]

Class: JobReschedulingService

Metric	Metrics Set	Description	Value	Regular Ra...
CHVL	Halstead M...	Halstead Volume	1115,7692	
CHD	Halstead M...	Halstead Difficulty	70,2963	
CHL	Halstead M...	Halstead Length	177	
CHEF	Halstead M...	Halstead Effort	78434,4417	
CHVC	Halstead M...	Halstead Vocabulary	79	
CHER	Halstead M...	Halstead Errors	0,6108	
WMC	Chidamber...	Weighted Methods Per Class	18	[0..12)
DIT	Chidamber...	Depth Of Inheritance Tree	1	[0..3)
RFC	Chidamber...	Response For A Class	40	[0..45)
LCOM	Chidamber...	Lack Of Cohesion Of Methods	1	
NOC	Chidamber...	Number Of Children	0	[0..2)
NOA	Lorenz-Kid...	Number Of Attributes	3	[0..4)
NOO	Lorenz-Kid...	Number Of Operations	19	
NOOM	Lorenz-Kid...	Number Of Overridden Methods	0	[0..3)
NOAM	Lorenz-Kid...	Number Of Added Methods	6	
SIZE2	Li-Henry M...	Number Of Attributes And Methods	22	
NOM	Li-Henry M...	Number Of Methods	7	[0..7)
MPC	Li-Henry M...	Message Passing Coupling	50	
DAC	Li-Henry M...	Data Abstraction Coupling	3	
NCSS	Chr. Cleme...	Non-Commenting Source Statements	36	[0..1000)
CMI	Maintainab...	Maintainability Index	32,5323	[0,0..19,0]

Evaluation of refactoring:

Overall the refactoring was a success by making the class more modular and robust, by splitting the functionality into smaller classes that have better metrics.