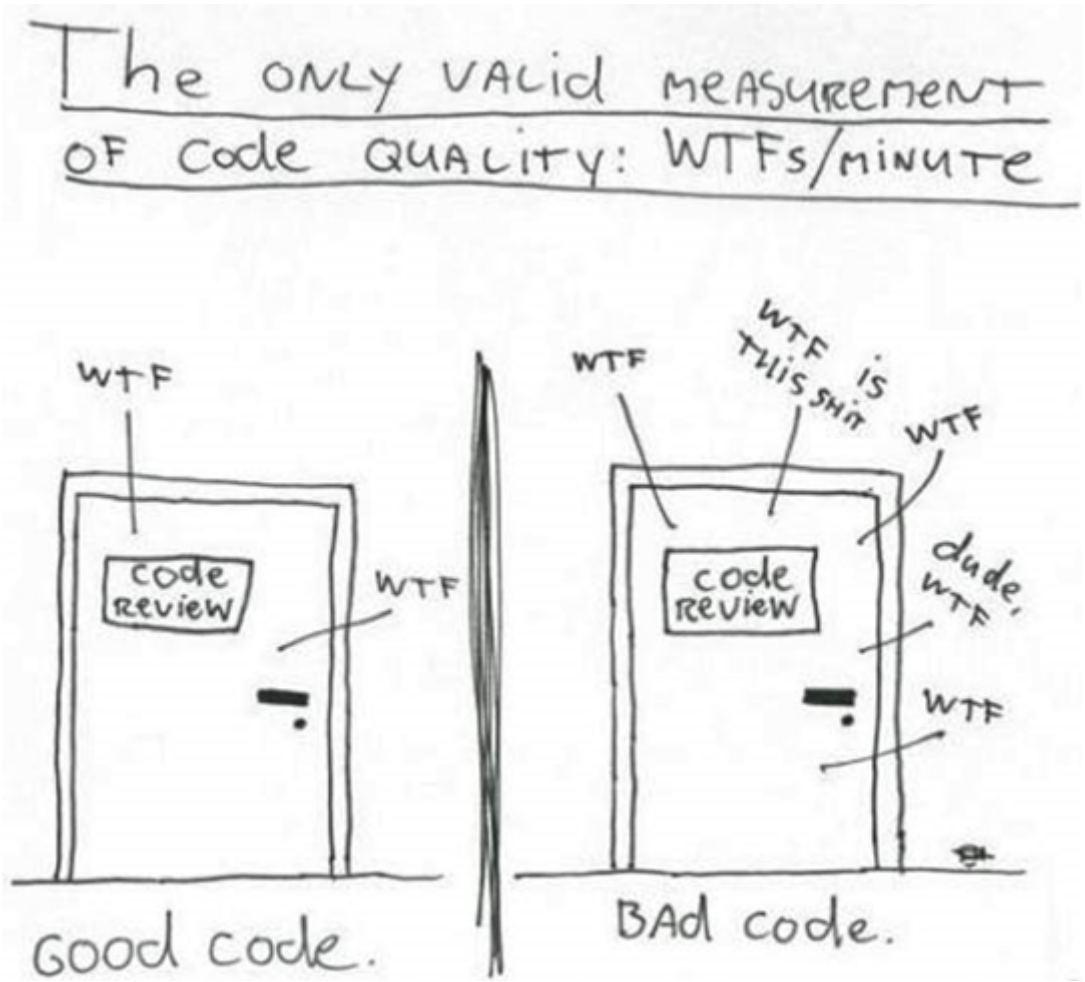
Assignment 2 - Refactoring

Group 13B
15-01-2022



Authors

A.F. Yücel	J.D.M. Savelkoul	M. Mladenov
M.C.A. de Wit	M.A.A. Kienhuis	W. Smit

Supervisor

George Hellouin de Ménibus

Comic via https://www.osnews.com/comics/

Part 1 Determining code smells

compute code metrics at the method
and/or class level (depending on the metric computed by the selected tool) and identify
which of them need to be improved by specifying and motivating the thresholds that allow
you to identify them

Class level smells (5x)

Controller functions now all individually deal with exceptions; they should be wrapped such that their wrapper deals with the specific mapping of exceptions to error status codes. This would remove a ton of duplicate code and improve cohesion.

- *<all controllers>.java*
    - Currently every method in a controller deals with returning specific error codes themselves. This leads to a lot of code duplication within and between the classes. This functionality can be extracted.
- *ta-microservice/src/main/java/nl/tudelft/sem/tams/ta/services/ContractService.java:*
    - This class suffers from low cohesion due to not properly using its own internal methods such as save and many class scoped variables. The usage of these methods should be implemented and common behaviour should be extracted.
- *hiring-microservice/src/main/java/nl/tudelft/sem/tams/hiring/services/HiringService.java:*
    - This class has medium-high coupling
- *hiring-microservice/src/main/java/nl/tudelft/sem/tams/hiring/services/ HiringService.java:*
    - This class has a medium-high Lack of Cohesion and thus low cohesion. The method pairs of this class should all share at least one attribute for an optimal cohesion.
- *hiring-microservice/src/main/java/nl/tudelft/sem/tams/hiring/controllers/HiringController.java:*
    - This class itself isn't necessarily too long, but the test suite is an excessive amount of lines in an order that doesn't make sense. Maybe this could be split into multiple controller classes in order to make the test class of a higher quality.
- *hiring-microservice/src/main/java/nl/tudelft/sem/tams/hiring/services/HiringService.java:*
    - This class has got a low-medium complexity. By replacing the for-loop in *getApplicationFromStudent(String netId)* by a stream, the complexity might go down even more.

Additional remarks

- *course-microservice/src/main/java/nl/tudelft/sem/tams/course/services/exceptions/ ConflictException.java:*
    - This class has medium-high complexity. The root class of all exceptions is Throwable and it implements the Serializable interface which requires the presence of a serialVersionUID. For the ConflictException class, serialVersionUID is what's causing this class to be labeled as complex, which in this case is impossible to get rid of due to PMD requirements.

Method level smells (5x)

- *hiring-microservice/src/main/java/nl/tudelft/sem/tams/hiring/services/HiringService.java: checkAndSave* - the McCabe complexity of this method is high

- *ta-microservice/src/main/java/nl/tudelft/sem/tams/ta/services/ContractService.java: createUnsignedContract* - the number of arguments of this method is too high; it would be better to combine those input variables into a model.

- *ta-microservice/src/main/java/nl/tudelft/sem/tams/ta/services/ContractService.java: createUnsignedContract* - the McCabe complexity of this method is high and the checks can be extracted into (a) separate method(s)

- *<multiple microservices>/AuthFilter.java: doFilterInternal* - the McCabe complexity of this method is high and functionality can be extracted into (a) separate method(s)

- *ta-microservice/src/main/java/nl/tudelft/sem/tams/ta/interfaces/CourseInformation.java: getCourseById* - this method returns an excessive amount of data that most of its callers do not need. courseInformation should expose methods for separate variables. CourseInformationResponseModel should never leave the course info service.

- *hiring-microservice/src/main/java/nl/tudelft/sem/tams/hiring/controllers/HiringController.java: apply(), getPendingApplications() & getRecommendedApplications():* These methods call a lot of other methods and might have to be split up into separate methods in order to make the code less complicated.

Part 2 Resolving code smells

Class level smells (5x)

- *hiring-microservice/src/main/java/nl/tudelft/sem/tams/hiring/services/HiringService.java*
    - This class had a low-medium complexity as described by codeMR and more specifically a calculated value of 26 for the Weighted Method Count (WMC). After changing our for loops to streams the WMC of this class went down from 26 to 22. In doing so we also lowered our coupling. With the code no longer relying on the outside classes such as ArrayList the CBO of affected methods was reduced by 2.

    - Secondly we were also interested in bringing down the coupling of this class overall. In combination with the refactorizations of the methods inside this service we have lowered overall CBO, similar to how it was performed in ContractService.java. This was done by moving external communication models out of the service and hiding them behind an interface. Combining these refactorizations with the aforementioned changes in mind we have managed to lower the overall class CBO by 2

- hiring-microservice/src/main/java/nl/tudelft/sem/tams/hiring/controllers/HiringController.java
    - This class was very long - 235 lines. It was split into two smaller controllers - one for dealing with applicant-oriented endpoints and one for handling endpoints used by lecturers, of 105 and 133 lines respectively. The common internal methods were moved to an abstract class inherited by both.

- *<all controllers>.java*
    - All controllers were returning specific error codes depending on the exception thrown, themselves. This leads to a lot of code duplication within and between the classes. Now, if an exception is thrown inside any of the methods in a controller corresponding to the same error code, it is being handled by an ExceptionHandler method.
      This resulted in a slightly lower LOC within each controller but more importantly improved overall consistency and removed boilerplate code.

- *ta-microservice/src/main/java/nl/tudelft/sem/tams/ta/services/ContractService.java*
    - This service was slowly falling into a state of decay with multiple helper methods that were either only used once or in the wrong class. We've moved class scoped variables into their respective functions wherever possible. We've removed overloaded methods that weren't used, moved the method for sending emails entirely to the email sender and improved cohesion by allowing findCourseBy to now take a null for netId as well, allowing this method to be re used more effectively by other parts of the code. These actions have resulted in a Tight lack of cohesion (LTCC) of 0 starting from 0.6 and a slightly lower LCOM, going from 0.8 to 0.6

Method level smells (5x)

- *ta-microservice/src/main/java/nl/tudelft/sem/tams/ta/services/ContractService.java: createUnsignedContract*
    - This method had 5 input parameters which may lead to mistakes of the order of the parameters. This got refactored by putting the parameters inside of a model. Now the method only has 1 input parameter.

- *hiring-microservice/src/main/java/nl/tudelft/sem/tams/hiring/services/ HiringService.java: checkAndSave*
    - According to the CK tool by M. F. Aniche, the McCabe complexity of this method was 5. After refactoring it by extracting the checks related to application and course status, the complexity was successfully reduced to 1. This refactoring improves code maintainability by making it easier both to understand its function and to reuse those checks where necessary.

- *ta-microservice/src/main/java/nl/tudelft/sem/tams/ta/services/ContractService.java: createUnsignedContract*
    - The McCabe complexity of this method was 7 according to the CodeMR tool.
      This method was also on the long side with 38 lines of code. After refactoring this was reduced to 21 lines of code by separating and isolating pieces of code inside of the original method into separate methods. The McCabe Complexity has been reduced to 1. This refactoring improved readability by reducing the amount of lines and reusability by exposing two methods that might need to be reused in the future.

- *hiring-microservice/src/main/java/nl/tudelft/sem/tams/hiring/controllers/ HiringController.java: apply(), getPendingApplications() & getRecommendedApplications():*
    - These methods were calling a lot of other methods initially (8, 7 and 12 respectively), making the code unnecessarily difficult and highly dependent on other methods. By extracting a few common parts and moving some logical functionality to the hiringService these numbers have been reduced to respectively 6, 3 and 4, making the methods easier to read and test.
- *ta-microservice/src/main/java/nl/tudelft/sem/tams/ta/interfaces/ CourseInformation.java: getCourseById()*
    - This method was returning an excessive amount of data. It has now been removed from the courseInformation interface and smaller more concrete methods have been added in its place such as getNumberOfStudents. In the actual implementation this still relies on getCourseById but now hides the communication object behind the interface. Lowering this method's CBO by 1 and helping us if we ever need to touch on our communication with the course microservice.
      This leads to a lack of tight cohesion within the ConnectedCourseInformation service but this will balance out overtime as more functions are added.

Metrics

Metrics generated via the CK tool by M.F. Aniche.

Before

Class metrics
https://gitlab.ewi.tudelft.nl/cse2115/2021-2022/sem-group-13b/sem-repo-13b/-/blob/69ad3220b44692e03cd3528749e604d6a679655c/docs/Code-Metrics/Before-Assignment-2/class.csv

Method metrics
https://gitlab.ewi.tudelft.nl/cse2115/2021-2022/sem-group-13b/sem-repo-13b/-/blob/69ad3220b44692e03cd3528749e604d6a679655c/docs/Code-Metrics/Before-Assignment-2/method.csv

After

Class metrics
https://gitlab.ewi.tudelft.nl/cse2115/2021-2022/sem-group-13b/sem-repo-13b/-/blob/69ad3220b44692e03cd3528749e604d6a679655c/docs/Code-Metrics/After-Assignment-2/class.csv

Method metrics
https://gitlab.ewi.tudelft.nl/cse2115/2021-2022/sem-group-13b/sem-repo-13b/-/blob/69ad3220b44692e03cd3528749e604d6a679655c/docs/Code-Metrics/After-Assignment-2/method.csv

Reflection

Currently when using only the codeMR visuals you will see that our lack of cohesion actually went up within the ta-microservice controllers. The reason for this is their newly added exception handlers which avoided a bunch of duplicate code. But sadly it counts as a method when it comes to cohesion measurements. The colour flip is nothing to be alarmed by as it is relative to a 0.02 increase in the actual underlying LCAM metric and .15 in the LTCC metric, due to controller methods being used by the Spring framework and being public due to that despite not being directly called in any other method.
Above stats are for the ContractController.java specifically but reasoning applies to all controllers.

We are happy with the refactorizations we have done but also recognize there is still space to improve. We would like to have our ta-microservice controllers also inherit from a single base controller similar to the hiring-microservice. We would like to also switch to using custom exceptions, as of right now there is a very high amount of coupling between both controller and its respective service due to the way exceptions are thrown and caught, but this is not detected by our code metrics. Finally we would like to take a closer look into all our controllers and services to divide them into smaller sections, we initially split them based on rough user story functionality and called it a day, certain controllers and services have now matured and are ready to be split on functionality again.
All of these could sadly not be looked into due to time constraints.