

# Exam CSE2115

## Software Engineering Methods

Delft University Of Technology  
January 28, 2022 - 9:00-11:00

Please, read these instructions carefully. The exam consists of **7** open questions that cover the material of the course. The questions are weighted and the weights (points) can be found in the header of each question. The total number of points that can be achieved on the exam is **50** points.

The exam has in total 11 pages.

---

### 1. DDD - Domain Driven Design (10 pts)

Consider the following scenario:

*“The Department of Software Technology at TU Delft wants to improve the students’ learning experience outside of class. For that reason, the development of an **online discussion forum** is envisioned to help students to review material before an assignment or exam, engaging students in a discussion of the course material before coming to class, and reflecting on material that they have read or worked with outside of class.*

*All users of the system need to authenticate themselves to determine who they are and what they can do on the platform. Users need to have a NetID associated with their account to identify them across the systems of the TU Delft universally. This NetID, together with a password, serves as the credentials for the account. The password needs to be stored safely. For simplicity, the system will not be connected to the existing TU Delft authentication system (Single-Sign-On). The NetID is just a unique string used to identify each user.*

*The forum system should have a different board (page) for each topic. These topics can, for example, be different courses or different parts of a course. Each topic can have multiple threads consisting of posts. Each thread starts with a question or statement and can have multiple posts replying to it. Topics can only be created by teachers. Threads and posts can be created by both teachers and students. Anonymous users should be able to access the forum and read threads; however, they shouldn't be able to create or post anything.*

*Forum topics have a name and description that is given by the users that create them. Threads have a title displaying the purpose of the thread, a body containing the question or statement, the creator of the thread, and the time it was created. Each post has the user creating the post, the time it was posted, and the body of the post. Topics, threads, and individual posts should all be uniquely linkable so that users can share links to these resources with other people.*

*Teachers are able to lock and unlock a topic or thread to prevent users from adding new content. If a topic or thread is locked, it is still visible to all users. However, users should be able to edit their topics, threads, and posts. When a topic, thread, or post is edited, it should display that it is edited and when the last edit was made.*

**QUESTION 1:** What are the **bounded contexts** for the scenario above? Explain the bounded contexts you have identified using Domain-Driven Design (DDD), their responsibilities and how they are related to one another. For each context, please specify whether it is a core, generic, or supporting context. Please motivate your design choices over alternative ones.

## 2. Requirement Engineering (5 pts)

Consider the following scenario:

*“Alice and Bob want to develop a new system for managing the grades of the students at TU Delft as an alternative to OSIRIS. After discussing with the different stakeholders, Alice and Bob have identified the following requirements:*

- 1. Student shall pass a course if grade is 5.75 or higher*
- 2. The system should have a maximum response time of 0.50s*
- 3. Teachers shall be able to enter grades for students*
- 4. Students shall be able to see their grades and their passed courses*
- 5. Teachers shall not be able to get grades*
- 6. Password shall be stored safely*
- 7. Users shall be able to interact with a GUI*
- 8. Students shall be able to get a message after registered for an exam*
- 9. The system shall be compatible with multiple operating systems*
- 10. The GUI shall use a color palette that is color-blind friendly*
- 11. The system shall provide statistics about the grade distribution to teachers*
- 12. Courses shall have multiple categories of grades*
- 13. Users must be able to log in using their NetID and password.*
- 14. Students must be able to view their courses.*
- 15. Teachers must be able to create new courses with a course code + course name.*
- 16. The application's dependencies will be managed through Gradle.*
- 17. The implementation of the application shall have at least 75% code coverage.*
- 18. The system should be able to handle at least 4000 parallel sessions.*
- 19. Any teacher can give a grade to a student for an exam.*
- 20. A teacher shall be able to change a grade if a mistake was made.”*

**(2 pts) QUESTION 2.1:** Examine the requirements listed above. Classify the requirements into functional (FR) and non-functional (NFR) requirements. Also, highlight any ambiguity in the requirements and indicate what is missing to complete the requirement's specification. If any requirements appear to be in conflict with one another, document that too.

**(1 pt) QUESTION 2.2:** What are the stakeholders for the grade management system discussed above? For each stakeholder, justify your choice and how they would contribute to the requirement analysis process.

**(2 pts) QUESTION 2.3:** Provide four additional functional and four additional non-functional requirements that, according to your opinion (as a potential stakeholder), should be added to the grade management system.

### 3. Design Patterns (10 pts)

Consider the following scenario:

*“We want to implement an online system for building robots. Robots are complex cyber-physical systems that have multiple hardware and software components that are chosen by customers based on their preferences.*

*A robot has different types of physical sensors:*

- *camera sensors for vision (eyes)*
- *tactile sensors for detecting objects (hands)*
- *microphones for detecting sounds and speech (ears)*

*For each type of sensor, customers should be able to choose a specific instance from the catalog. For example, there are alternative sensors for the eyes (e.g., full HD camera, wide-angle camera, etc.)*

*Besides, the signals from each sensor are fed to a specific Artificial Intelligence (AI) engine, such as:*

- *Deep-learning based recognition for the camera*
- *Simple regression model for the hands*
- *Text-to-speech algorithm for the sounds*

*Using our system, customers should be able to build up a robot selecting the sensor and the AI engine they want incrementally.”*

**(4 pts) QUESTION 3.1:** Which design pattern fits best in the implementation of this system? Please explain your choice.

**(6 pts) QUESTION 3.2:** For the given design pattern of your choice (part 1), draw the corresponding class diagram containing the following elements:

- The classes (the role, class name, attributes, and method names, etc.) you would write to implement the design pattern
- The type of association for each pair of classes/interfaces.

## 4. Software Analytics 1 (5 pts)

Consider the program in Listing 1. Answer the following questions:

**(2 pts) QUESTION 4.1:** What is the value of the LCOM (Lack of Cohesion of Methods) metric for the `Fibonacci` class? Show all the steps you used to compute the LCOM (how you applied the formula).

**(3 pts) QUESTION 4.2:** Given the results of the LCOM, does the class contain a code smell? If yes, which one? How would you address the code smell?

Listing 1

```
1. public class Fibonacci {
2.
3.     private static Map<Integer, Integer> map = new HashMap<>();
4.
5.
6.     public static void main(String[] args) {
7.         Scanner sc = new Scanner(System.in);
8.         int n = sc.nextInt();
9.
10.        System.out.println(fibMemo(n));
11.        System.out.println(fibBotUp(n));
12.        System.out.println(fibOptimized(n));
13.        sc.close();
14.    }
15.
16.
17.    /* This method finds the nth fibonacci number using memoization
18.    technique */
19.    public static int fibMemo(int n) {
20.        if (map.containsKey(n)) {
21.            return map.get(n);
22.        }
23.
24.        int f;
25.
26.        if (n <= 1) {
27.            f = n;
28.        } else {
29.            f = fibMemo(n - 1) + fibMemo(n - 2);
30.            map.put(n, f);
31.        }
32.        return f;
33.    }
34.
35.
```

```

36.  /* This method finds the nth fibonacci number using bottom up */
37.  public static int fibBotUp(int n) {
38.      Map<Integer, Integer> fib = new HashMap<>();
39.      for (int i = 0; i <= n; i++) {
40.          int f;
41.          if (i <= 1) {
42.              f = i;
43.          } else {
44.              f = fib.get(i - 1) + fib.get(i - 2);
45.          }
46.          fib.put(i, f);
47.      }
48.
49.      return fib.get(n);
50.  }
51.
52.  /** This method finds the nth fibonacci number using bottom up */
53.  public static int fibOptimized(int n) {
54.      if (n == 0) {
55.          return 0;
56.      }
57.      int prev = 0, res = 1, next;
58.      for (int i = 2; i <= n; i++) {
59.          next = prev + res;
60.          prev = res;
61.          res = next;
62.      }
63.      return res;
64.  }
65.  }

```

## 5. Software Analytics 2 (5 pts)

Consider the program in Listing 2. Answer the following questions:

**(3 pts) QUESTION 5.1:** What are the decision points for the `divideMessageWithP` method?

**(2 pts) QUESTION 5.2:** What is the Cyclomatic Complexity (CC) for the `divideMessageWithP` method?

Listing 2

```
1. public class CRCAlgorithm {
2.
3.     private int correctMess;
4.     private int wrongMess;
5.     private int wrongMessCaught;
6.     private int wrongMessNotCaught;
7.     private int messSize;
8.     private double ber;
9.     private boolean messageChanged;
10.    private ArrayList<Integer> message;
11.    private ArrayList<Integer> dividedMessage;
12.    private ArrayList<Integer> p;
13.    private Random randomGenerator;
14.
15.    /**
16.     * The algorithm's main constructor. The most significant variables,
17.     * used in the algorithm, are set in their initial values.
18.     * @param str The binary number P, in a string form, which is
19.     * used by the CRC algorithm
20.     * @param size The size of every transmitted message
21.     * @param ber The Bit Error Rate
22.     */
23.    public CRCAlgorithm(String str, int size, double ber) {
24.        messageChanged = false;
25.        message = new ArrayList<>();
26.        messSize = size;
27.        dividedMessage = new ArrayList<>();
28.        p = new ArrayList<>();
29.        for (int i = 0; i < str.length(); i++) {
30.            p.add(Character.getNumericValue(str.charAt(i)));
31.        }
32.        randomGenerator = new Random();
33.        correctMess = 0;
34.        wrongMess = 0;
35.        wrongMessCaught = 0;
36.        wrongMessNotCaught = 0;
37.        this.ber = ber;
38.    }
39.
40. }
```

```

41.  /* The most significant part of the CRC algorithm. */
42.  public void divideMessageWithP(boolean check) {
43.      ArrayList<Integer> x = new ArrayList<>();
44.      ArrayList<Integer> k = (ArrayList<Integer>) message.clone();
45.      if (!check) {
46.          for (int i = 0; i < p.size() - 1; i++) {
47.              k.add(0);
48.          }
49.      }
50.      while (!k.isEmpty()) {
51.          while (x.size() < p.size() && !k.isEmpty()) {
52.              x.add(k.get(0));
53.              k.remove(0);
54.          }
55.          if (x.size() == p.size()) {
56.              for (int i = 0; i < p.size(); i++) {
57.                  if (x.get(i) == p.get(i)) {
58.                      x.set(i, 0);
59.                  } else {
60.                      x.set(i, 1);
61.                  }
62.              }
63.              for (int i = 0; i < x.size() && x.get(i) != 1; i++) {
64.                  x.remove(0);
65.              }
66.          }
67.      }
68.      dividedMessage = (ArrayList<Integer>) x.clone();
69.      if (!check) {
70.          for (int z : dividedMessage) {
71.              message.add(z);
72.          }
73.      } else {
74.          if (dividedMessage.contains(1) && messageChanged) {
75.              wrongMessCaught++;
76.          } else if (!dividedMessage.contains(1) && messageChanged) {
77.              wrongMessNotCaught++;
78.          } else if (!messageChanged) {
79.              correctMess++;
80.          }
81.      }
82.  }
83.
84.  ...
85.  }

```



## 6. Regression Testing (5 pts)

Consider the coverage matrix (rows are test cases and columns are branches) depicted in Table 1.

**QUESTION:** Which test cases are selected when using the additional greedy algorithm to reach 100% branch coverage?

Report the test case(s) in the exact order they are selected by the additional greedy algorithm. Also indicate the branches that are additionally covered by the test case selected in each iteration of the algorithm.

**Table 1**

	b1	b2	b3	b4	b5	b6	b7	b8	b9	b10	b11	b12	b13	b14	b15	b16	b17	b18	b19	b20
t1			x							x	x				x			x		
t2		x		x				x					x	x					x	
t3	x		x	x					x			x			x					
t4			x		x	x	x			x									x	
t5							x	x						x				x	x	
t6		x	x								x			x						x
t7							x							x	x	x				x
t8		x	x	x				x	x	x								x		
t9	x		x	x	x					x	x									
t10				x	x	x	x											x	x	
t11							x	x			x	x								x
t12			x	x	x								x		x	x				
t13																	x	x	x	x
t14		x	x						x	x				x	x					
t15	x																x	x	x	x

## 7. Mutation Testing (10 pts)

Consider the program in Listing 3. Let us consider the mutant where line **43** (third condition of the if statement) is changed as follows:

- Original condition: `string.contains(banned_digits[2])`
- Mutated condition: `string.contains(banned_digits[0])`

**Question:** Write a test case that can kill the mutant. Remember that a meaningful test case must include assertions. Syntax errors (e.g., missing semicolons, spelling mistakes, etc.) will be ignored.

Listing 3 (Original Program)

```
1.  class Solution {
2.      int A = 1;
3.      int B = 3;
4.      int C = 9;
5.
6.      private String[] banned_digits;
7.
8.      public Solution() {
9.          banned_digits = new String[]{
10.              Integer.toString(A),
11.              Integer.toString(B),
12.              Integer.toString(C)
13.          };
14.      }
15.
16.      /**
17.       * This method counts the numbers from 0 to n with repeated digits.
18.       * For example, the only positive number (n<= 20) with at least 1
19.       * repeated digits is 11.
20.       * However, this method has a special tweak: all numbers that
21.       * contains the digits in A, B, or C are not counted.
22.       * @param n
23.       */
24.      public int numbers_duplicate_digits(int n) {
25.          if (n<=0)
26.              return 0;
27.
28.          int counter = 0;
29.          for (int i=0; i<=n; i++){
30.              String s = Integer.toString(i);
31.              if (this.hasDuplicates(s)) {
32.                  counter++;
33.              }
34.          }
35.
36.          return counter;
37.      }
```

```

38.
39.     public boolean hasDuplicates(String string){
40.         // let's handle the special cases
41.         if (string.contains(banned_digits[0]) ||
42.             string.contains(banned_digits[1]) ||
43.             string.contains(banned_digits[2])) <-- MUTANT LOCATION
44.             return false;
45.
46.         // let's handle all the other cases
47.         for (int i = 0; i < string.length(); i++){
48.             char current = string.charAt(i);
49.             String newString = string.replaceAll(current+"", "");
50.             if (newString.length() < string.length()-1)
51.                 return true;
52.         }
53.
54.         return false;
55.     }
56. }

```

**THIS IS THE END OF THE EXAM.**