Exam CSE2115
Software Engineering Methods

Delft University Of Technology
April 4, 2022 - 9:00-12:00

Please, read these instructions carefully. The exam consists of **6** 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 **60** points.

The exam has in total **8** pages.

---

1. DDD - Domain Driven Design (10 pts)

Consider the following scenario:

"*TU Delft needs a new system for the **library**. For that reason, the development of an online **booking system** for the **library** is envisioned to help students and teachers access the catalog of books and multimedia education material available in the library.*

Teachers and Students (hereafter referred to as users) can access the library catalog using their NetID. Once logged in, users can browse the catalog, search for the library items (e.g., books) via keyword search or by browsing the catalog. Once the user finds the item to reserve, they can reserve the chosen item. The users have to choose the pick-up time and location for physical items (books, CDs, and DVDs). In case there is no copy available for the chosen physical items, the reservation is set in the 'waiting' state, and the library will notify the user (via email) when the item is available to collect.

Audiobooks and ebooks are available online and are immediately available to the users (i.e., no need to check for availability and collect time).

Users can cancel a reservation at any time from their account. Users can also browse their reservation history and current reservation status. A reservation has an ID, the

user's name who made the reservation, and the item's ID. A physical item cannot be booked for more than one month.

The system should also manage a reputation scoring system for the users. A user that reports an item late or damaged receives one negative point (negative reputation). If the item is returned in time and in a good state, the user receives a positive point (positive reputation). Users with a very negative reputation (-5 points) are no longer allowed to make a reservation. Finally, the system has an administrator that can add more library items and reset (to zero) the users' reputation scores."

QUESTION 1: What are the **bounded contexts** for the scenario above? Explain the bounded contexts you identified using Domain-Driven Design (DDD), their responsibilities, and how they are related.

For each context, please specify whether it is a core, generic, or supporting context. Please motivate your design choices over **alternative** ones.

REMARK: There is no page/word limit for your answer. However, we highly recommend writing concise answers that contain the following elements:

a. The name and the responsibility of each bounded context (1-2 sentences per context) and their classification (core, generic, etc.).
b. Interaction among the identified bounded contexts (1 sentence per pair of bounded contexts).
c. Motivation (1-2 sentences).
d. Alternative (1-2 sentences per alternative).

2. Requirement Engineering (10 pts)

Consider the following scenario:

"*TU Delft wants to improve the education experience for both teachers and students. Therefore, the university hired our development team to develop a novel system for online education, and that allows students to attend lectures online, submit live questions, participate in online polls, etc. The system should also facilitate teachers in preparing and setting up interactive education tools (e.g., for live polls).*
The development team has identified the following requirements:

1. *The system should allow the responsible teachers to create live polls for a given lecture.*
2. *The system should allow students enrolled in a course to participate in active live polls.*
3. *The students should allow students that are enrolled in a course to submit questions related to the lecture content.*
4. *The system should allow users to authenticate themselves via the TU Delft Single-Sign-On.*
5. *The system should include profanity checks for the free-text questions and polls.*
6. *The system should be developed in Java.*
7. *Students should not be able to add/delete/change polls.*
8. *Students should be able to post online questions.*
9. *The system should be GDPR-compliant.*
10. *The system should allow lecturers to record the lectures.*
11. *The system should store the lecture videos using the MPEG compression standard.*
12. *The system should allow the responsible teacher to indicate which live question will be answered next during the lecture.*
13. *The system should have a fast response time.*
14. *The system should be able to handle at least 1000 users per online lecture.*
15. *The data should be stored using SQLite version 3.8.*
16. *The system should allow students to replay the recorded lectures.*
17. *The system should not allow posting questions for already recorded lectures.*
18. *The system should allow teachers to add/delete/change polls.*
19. *The system should have a low energy consumption.*
20. *The system should allow administrators to add new users.*"

(4 pts) QUESTION 2.1: Examine the requirements listed above. Classify the requirements into functional (FR) and non-functional (NFR) requirements. Also, highlight any ambiguity (i.e., not clearly formulated) in the requirements and indicate what is missing to complete the requirement's specification. If any requirements conflict with one another, please document that too.

Notice that to answer this question, you can simply refer to the requirements number (hence, there is no need to rewrite the requirements).

(3 pts) QUESTION 2.2: What are the stakeholders for the online education tool discussed? Please indicate at least three stakeholders. For each stakeholder, justify your choice and how they would contribute to the requirement analysis process.

(3 pts) QUESTION 2.3: Provide four additional functional and two 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 a graphical editor that allows users to build complex diagrams out of simple graphical elements (such as lines, squares, text boxes, etc.). The user can group these elements to form large elements, which in turn can be grouped to form still larger elements (e.g., in UML diagrams).*

The system should allow drawing all elements of one single diagram at once, ***as well as*** *changing the characteristics of all its internal elements altogether (e.g., changing the color of the lines, boxes, and text in the diagram). In other words, changes to the diagram should be recursively applied to all its internal graphical elements.*"

QUESTION: Which design pattern fits best in the implementation of this system? Please explain your choice.

For the given design pattern of your choice, draw the corresponding class diagram containing the following elements:

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

4. Software Analytics (10 pts)

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

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

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 StringMatchFiniteAutomata {
2.
3.     public static final int CHARS = 256;
4.     public static int[][] FA;
5.     public static Scanner scanner = null;
6.
7.     public static void main(String[] args) {
8.         scanner = new Scanner(System.in);
9.         System.out.println("Enter String");
10.        String text = scanner.nextLine();
11.        System.out.println("Enter pattern");
12.        String pat = scanner.nextLine();
13.        searchPat(text, pat);
14.        scanner.close();
15.    }
16.
17.    public static void searchPat(String text, String pat) {
18.        int m = pat.length();
19.        int n = text.length();
20.
21.        FA = new int[m + 1][CHARS];
22.        computeFA(pat, m, FA);
23.
24.        int state = 0;
25.        for (int i = 0; i < n; i++) {
26.            state = FA[state][text.charAt(i)];
27.            if (state == m) {
28.              System.out.println("Pattern found at idx " + (i - m + 1));
29.            }
30.        }
31.    }
32.
33.    // Computes finite automata for the pattern
34.    public static void computeFA(String pat, int m, int[][] FA) {
35.        for (int state = 0; state <= m; ++state) {
36.            for (int x = 0; x < CHARS; ++x) {
37.                FA[state][x] = getNextState(pat, m, state, x);
38.            }
39         }
40.    }
41.
42.    public static int getNextState(String pat, int m, int state, int x) {
43.       // if current state is less than length of pattern and input
44.       // character of pattern matches the character in the alphabet
45.       // then automata goes to next state
46.        if (state < m && x == pat.charAt(state)) {
47.            return state + 1;
48.        }
49.
50.        for (int ns = state; ns > 0; ns--) {
51.            if (pat.charAt(ns - 1) == x) {
52.                for (int i = 0; i < ns - 1; i++) {
53.                   if (pat.charAt(i) != pat.charAt(state - ns + i + 1)) {
54.                        break;
55.                    }
56.
```

```
57.                     if (i == ns - 1) {
58.                         return ns;
59.                     }
60.                 }
61.             }
62.         }
63.         return 0;
64.     }
65. }
```

5. Regression Testing (10 pts)

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

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			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				x
t8			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

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.

6. Mutation Testing (10 pts)

Consider the program in Listing 2. Let us consider the mutant where line **14** (second statement in the method `hex2decimal` method) is changed as follows:

- Original version: `s = s.toUpperCase();`
- Mutated version: `s = s ;` (it corresponds to deleting line 14)

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 2 (Original Program)

```
1.  /**
2.  * Converts any Hexadecimal Number to Octal
3.  */
4.
5.  public class HexToOct {
6.
7.      /**
8.       * This method converts a Hexadecimal number to a decimal number
9.       * @param s The Hexadecimal Number
10.      * @return The Decimal number
11.      */
12.     private static int hex2decimal(String s) {
13.         String str = "0123456789ABCDEF";
14.         s = s.toUpperCase(); <-- MUTANT LOCATION
15.         int val = 0;
16.         for (int i = 0; i < s.length(); i++) {
17.             char a = s.charAt(i);
18.             int n = str.indexOf(a);
19.             val = 16 * val + n;
20.         }
21.         return val;
22.     }
23.
24.     /**
```

```
25.      * This method converts a Decimal number to a octal number
26.      * @param q The Decimal Number
27.      * @return The Octal number
28.      */
29.     private static int decimal2octal(int q) {
30.         int now;
31.         int i = 1;
32.         int octnum = 0;
33.         while (q > 0) {
34.             now = q % 8;
35.             octnum = (now * (int) (Math.pow(10, i))) + octnum;
36.             q /= 8;
37.             i++;
38.         }
39          octnum /= 10;
40.         return octnum;
41.     }
42.
43.     /**
44.     * Main method that gets the hexadecimal input from user and converts
45.     * it into octal.
46.     */
47.     public int convert(String hexadecimal) {
48.
49.         // convert hex to decimal
50.         int decnum = hex2decimal(hexadecimal);
51.
52.         // convert decimal to octal
53.         return decimal2octal(decnum);
54.     }
55. }
56.
```

THIS IS THE END OF THE EXAM.