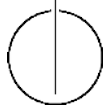




ASE Mock Exam blanked

Advanced Topics in Software Engineering (Technische Universität München)



Mock Examination: IN2309 - Advanced Topics of Software
Engineering - 11.02.2022

First Name:

Last Name:

Matr. No.:

Course of Studies:

I hereby declare that I agree to the above mentioned terms and feel mentally and physically fit to participate in the exam.

(Student Signature)

Question	1	2	3	4	5	6	7	8	9	Sum	Grade
Points											
Max. Points	11	10	11	9	11	16	13	12	7	100	

1 Distributed Systems

11 points

1. Name 4 characteristics of distributed systems.

(2 Points)

2. Briefly explain the differences between horizontal and vertical scaling.

(1 Point)

- Horizontal: add more machines with the same or different computing power.
- Vertical: in one machine, add more hardware or replace the existing hardware with a more powerful one.

3. Briefly explain the MapReduce algorithm and translate the following SQL statement to a MapReduce function

(5 Point)

```
1 SELECT employee FROM employees WHERE salary_level > 3
```

(2 points explanation 3 points MapReduce function)

- A MapReduce program is composed of a map procedure (or method), which performs filtering and sorting (such as sorting students by first name into queues, one queue for each name), and a reduce method, which performs a summary operation (such as counting the number of students in each queue, yielding name frequencies) (Wikipedia)

```
1 map(key, record):  
2     if(record.salary_level > 3)  
3         selection = (record.employee)  
4         emit(key, selection)
```

4. Briefly explain the differences between Data Marts and Data Cubes.

(3 Point)

- A data mart (DM) is a subset or an aggregation of the data stored to primary warehouse.
- Data cubes are aggregated facts based on chosen dimensions. Data cubes are three dimensional data structures.

2 Security

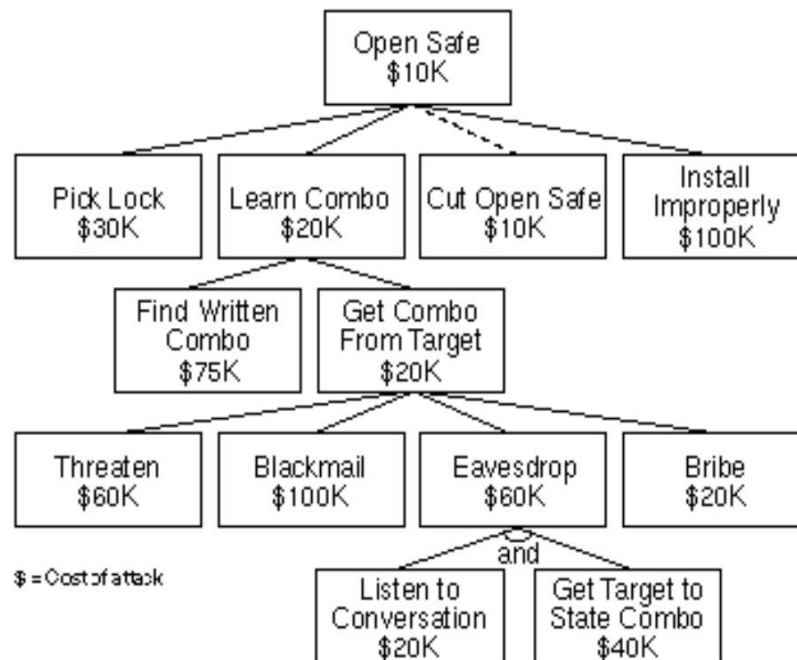
10 points

1. Briefly describe the difference between Security and Safety in software systems. (1 Point)

- Security — Protection of systems (products, devices) from external hazards
- Safety — Protection against hazards and risks originating from the operation of a system (product, device)

2. Briefly explain the least privilege principle and give an example for it. (2 Point)

3. Below you'll find an illustration of a potential attack tree. State the paths for the most expensive and the cheapest attack for opening the the save and state how costly each of the paths is. (3 Point)



4. You were hired by a local software development company as security consultant. As a first task, you should analyze the following programme:

```

1
2 public class UserCount {

```

```
3
4 public static boolean duplicateUser(String username) throws Exception {
5
6     // ...
7     String sqlStatement = "SELECT COUNT(*) FROM user_table where username = " +
8         username;
9     // ...
10    int count = jdbcConnection.execute(sqlStatement);
11    if(count <= 1) {
12        return true;
13    }
14    return true;
15 }
16
17 public class Main {
18
19     public void doSomething() {
20         // ...
21         // POST("ATTRIBUTE_NAME") receives unfiltered HTTP post request from the
22         // client
23         String username = POST("username")
24         UserCount.duplicateUser(username);
25         //..
26     }
27 }
```

code above

(2 points)

- SQL Injection
- An attack can inject some SQL statement into the POST("username") variable. For example, ' ABB' OR 1=1' -;

(b) Briefly explain how the vulnerability can be fixed

(2 points)

- Use prepared statements or escape the user inputs properly before creating the SQL statement string.

3 Anti Patterns

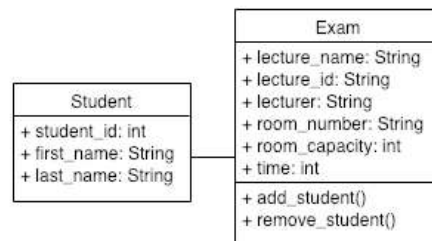
11 points

1. What are the "7 deadly sins" in software practice? Name **three** of them and describe each with one sentence. (3 points)

0.5p per name/0.5 per description

- Haste — Solutions based on hasty decisions ("time is most important") lead to compromises in software quality
- Apathy — Not caring about a problem, followed by unwillingness to attempt a solution
- Narrow-mindedness — The refusal to use solutions that are widely known ("Why reuse? I only have to solve one problem")
- Sloth — Making poor decisions based on "easy" answers
- Avarice — No use of abstractions, excessive modeling of details
- Ignorance — Failure to seek understanding
- Pride — Not invented here (NIH): Not willing to adopt anything from the outside

2. Analyse the following UML class diagram for an exam management system.



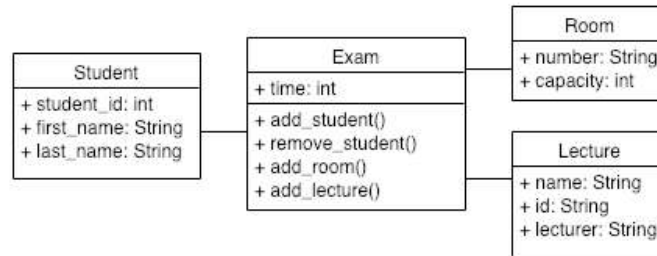
- (a) Which of the anti-patterns discussed in the lecture can you identify? Justify your answer.

(1 points)

0.5p name + 0.5 justification

The Blob.

- (b) Re-factor the design of the system and draw the new UML class diagram without the anti-pattern. (4 points)
1p per class



- (c) Name **three** typical causes for the anti-pattern you identified in (a). (3 points)
1p per cause

- Lack of an object-oriented architecture. The designers may not have an adequate understanding of object-oriented principles.
- Lack of (any) architecture. The absence of definition of the system components, their interactions, and the specific use of the selected programming languages.
- Lack of architecture enforcement. Sometimes this Antipattern grows accidentally, even after a reasonable architecture was planned. This may be the result of inadequate architectural review as development takes place.
- Too limited intervention. In iterative projects, developers tend to add little pieces of functionality to existing working classes, rather than add new classes, or revise the class hierarchy for more effective allocation of responsibilities.
- Specified disaster. Sometimes the Blob results from the way requirements are specified. If the requirements dictate a procedural solution, then architectural commitments may be made during requirements analysis that are difficult to change.

4 Blockchain-based architectures

9 points

1. Name **two** characteristics of smart contract platforms.

(2 points)

1p per characteristic

2. Describe the role of a wallet owner in a blockchain network as described in the lecture. (3 points)

1p per point

3. Name **two advantages** and **two disadvantages** of blockchain-based systems compared to centralized platforms. (4 points)

1p per point

Advantages:

5 Software Architecture

11 points

Dependency Structure Matrix (DSM) is a compact way to express dependencies in a software project. It helps developers to understand components that belong together and suggests a way to partition a program.

2. Given the following java class.

```

1 public boolean transferMoney(Account sourceAccount, Account targetAccount,
   BigDecimal moneyAmount) {
2     sourceAccount.transfer(targetAccount, moneyAmount);
3     return true;
4 }

```

Under what circumstances might this function fail? Write suitable two pre-conditions and one post- condition for this Java function. (4 points)

Answer

1
2
3
4
5
6
7

3. You are working as a software developer in GameForLife company. They are developing a new multiplayer online role-playing game (MMORPG) based on real life birds. Your colleague developed the following interface as a base for birds:

```

1 public interface Bird {
2     public boolean eat(){ }
3     public boolean fly(){ }
4     public boolean breed(){ }
5 }

```

Afterwards you finished your implementation for both eagle and penguin and submit it as follows:

```

1 public class Eagle implements Bird {
2     public boolean eat(){
3         ....
4         // Some code
5         ....
6     }
7     public boolean fly(){
8         ....
9         // Some code

```

```
10     ....
11     }
12     public boolean breed(){
13         ....
14         // Some code
15         ....
16     }
17 }
18
19 public class Penguin implements Bird {
20     public boolean eat(){
21         ....
22         // Some code
23         ....
24     }
25     public boolean fly(){
26         ....
27         // Some code
28         ....
29     }
30     public boolean breed(){
31         ....
32         // Some code
33         ....
34     }
35 }
```

Later on your boss came and asked you the following questions:

- (a) Is Eagle and Penguin a valid Java implementation of *Bird*? Why/why not? (2 point)

Answer

- (b) Is the implementation consistent with Liskov's substitution principle? If yes, why? If not, how would you make it consistent by writing a correct java implementation? (4 points)

Answer

6 Testing

16 points

1. Explain briefly what is the *testability* of a program. (2 point)



Figure 3: What is testability? (2)

2. The Storage-For-You company provides storage renting rooms for customers in various locations in Germany. As a tester for the company you are given the task of testing their storage processing system. Assume that a storage room record looks like this:

- ID: int from 1 - 10000.
- address: String [a-zA-Z0-9\s] with exactly 30 characters.
- price: float, that varies from 5 euro per day to 20 euro per day. Rates are multiples of 0.5 euro.
- size: float, that ranges from 1 to 30 meter squares. Sizes are multiples of 1 meter squares.

- (a) How many possible types of records (and thus test cases) are there? (3 point)

Answer




- (b) Assume the execution of each test case takes 2ms,
i. Is it feasible to completely test this program? Justify your answer! (1 point)

Answer



- ii. How could you reduce the *input space* test the program without having issues later ?
If you have a continues integration server that runs all test cases every night (assume a night lasts 11 hours) can you test all your test cases or do you need more servers ? (4 point)

Answer

3. Consider the following Java code to find the factorial of a number .

```
1 static int factorial(int n){  
2     if (n == 0)  
3         return 1;  
4     else  
5         return(n * factorial(n-1));  
6 }
```

- (a) Write the JUnit tests for it "Assuming it should return 0 if there is an error". (2 points)



- (b) Did the code pass all your tests or not ? If yes, why? If no, explain possible fail cases? (1 points)

Answer

- (c) Can you completely test this factorial function ? Justify your answer. (1 points)

Answer

In theory you could test all values from 0 to Integer.MAX_VALUE. In practice this is just not feasible, because recursion function is time consuming process and we can't execute all of these checks in reasonable amount of time.

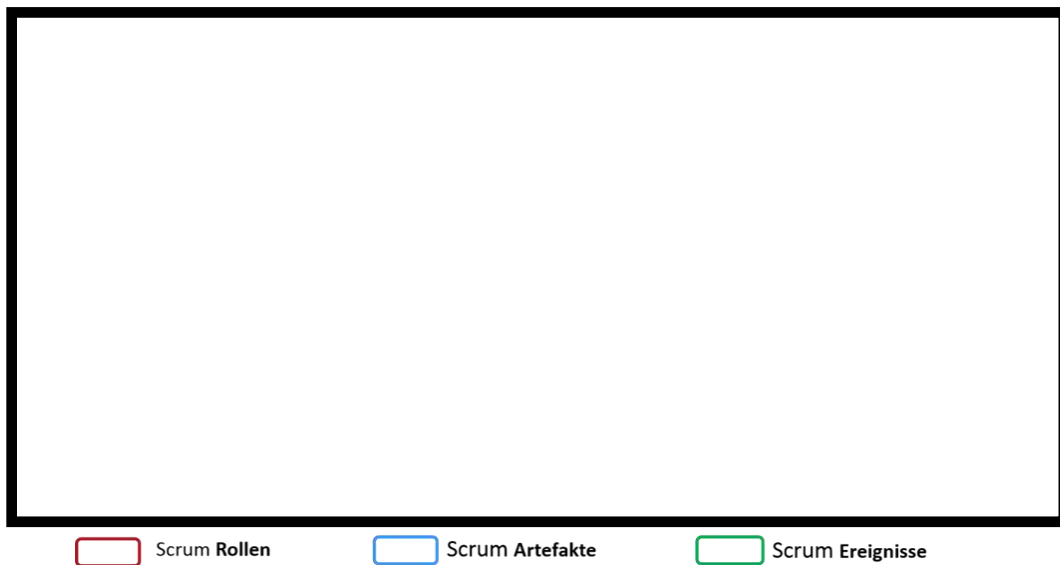
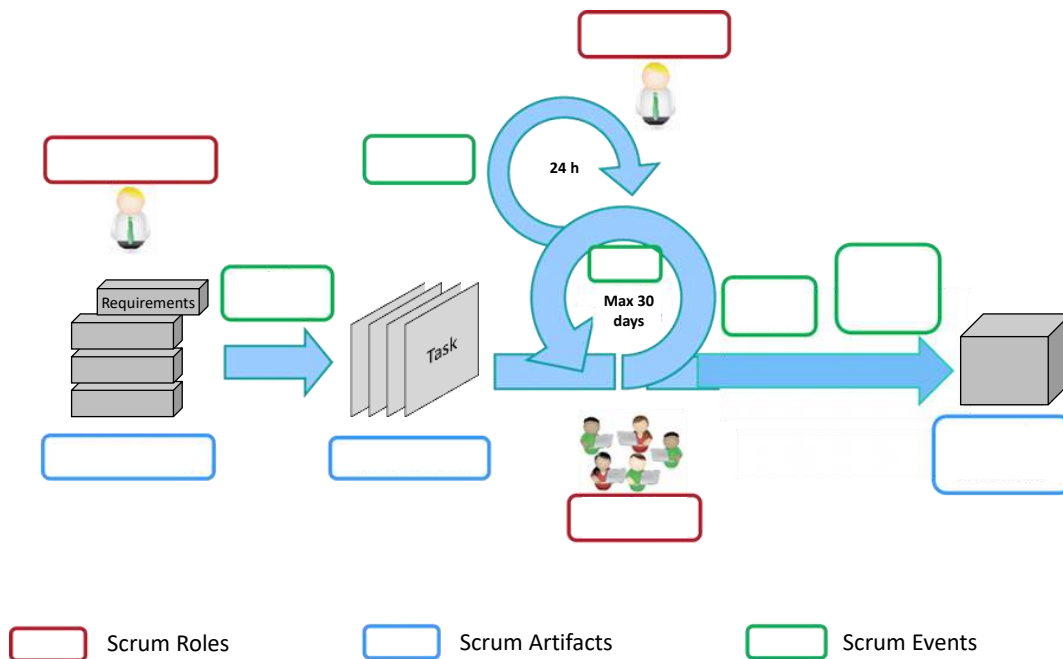
- (d) Give two reasons that could cause a system to behave in a non-deterministic way during testing (i.e. flaky tests)? (3 points)

Answer

7 Scrum and Software Quality

12 points

1. Fill in the missing artifacts in the schema of the Scrum Agile Methodology process. (6 points)



2. Name and briefly explain the scrum roles. (3 points)

3. Name and explain four software quality factors.

(4 points)

8 Coupling and Cohesion

12 points

1. Explain the difference between coupling and cohesion. (2 points)

2. Explain the relationship of coupling and cohesion of a bad system architecture. (2 points)

3. Consider you are a software architect in a car manufacturing company. Look at the design of a car driving behavior analyzer, *iCarBehave*, and its related components in Fig. 4.

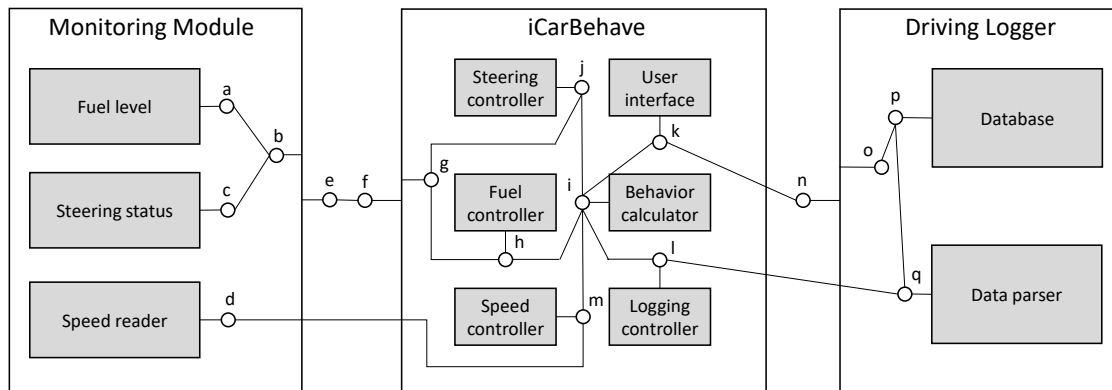


Figure 4: Car driving behavior analyzer, *iCarBehave*

Answer questions on the next page related to this design.

- (a) Briefly explain 2 lowest coupling levels.

(4 points)

Solution

- (b) List at-least 4 *changes* that you can make in this system's design to decrease the coupling factor. (2 points)

Note: Example of one change may be removing a connection or adding a connection.

Solution

- (c) Name and explain the architecture principle that is violated on this example.

(2 points)

Solution

9 Quiz

7 points

1. Evaluate the following statements (correct answers + 1 point, incorrect answers -1 point, you can not get less than 0 points overall): (7 Points)

Statement	True	False
Smart contracts guarantee correctness of code.		X
An embedded system is an information system integrated into a physical thing.	X	
Product Line Engineering increases development costs.		X
Continuous integration makes it more difficult to detect integration bugs.		X
Stateless EJBs persist over multiple sessions.		X
REST is language agnostic..	X	
In Autonomous Vehicle Testing, Acceptance Test verifies if the implementation meets business or functional requirements.	X	