Exam CS2115
Software Engineering Methods

Delft University Of Technology
January 29, 2021 - 9:00-11:00

Please, read these instructions carefully. Failure to comply with any of the instructions means invalidating your exam.

The exam consists of six open questions that cover the context of the course. Be aware that the exam questions are designed to prevent plagiarism and student collaborations using variants and by introducing subtle differences across the variants. We will also use plagiarism detection tools to identify possible plagiarized text and potential frauds.

Potential plagiarism and frauds will be reported to the Board of Examiners.

---

1. Consider the following scenario (Carpooling Management System):
   In 2020, a group of 3 recent graduate students from the Master in Computer Science at TU Delft decided to create a startup, aiming to provide high-tech digital services using cutting-edge software technology.

   Amtech is an American company interested in giving a carpooling service in the Netherlands and hired the startup to develop CarPSys, a Car Pooling Management System. The solution consists of two parts, a management platform for monitoring the service and a mobile application (for drivers and consumers). During the first stage of the project, the goal is to develop a working prototype (for the entire solution) and launch a mobile application only for people within the Randstad area (four largest Dutch cities).

   Amtech's CEO, Ellie Martin, wants to evaluate if the solution is scalable and profitable. Since most people in the Netherlands care about preserving the environment, there is an excellent first impression of the project's sustainability.

However, other managers in Amtech are sceptical about the viability of the project. They want to have access to financial reports to identify the profitability of the carpooling service.

The Customer Relationship (CR) manager's primary concern is on delivering a reliable service. Collecting feedback from consumers about the service is a priority for his team. Additionally, CarPSys's customers should reach Amtech through its Contact Center (CC) platform, which unifies all communication channels, i.e., phone calls, text messages, social media messages, etc. The CC manager should ensure that the information regarding customer inquiries and answers is preserved and available to other departments in Amtech who might need it.

Maarten Boekema, Amtech's CTO (Chief Technical Officer), has assumed the project's responsibility and is the main point of contact between the startup and Amtech. Under some pressure from Ellie and other managers, and after several discussions with them, Maarten has started identifying the system's requirements but has quickly got out of his depth. With a proven background in business analysis and software engineering, he has now hired you to continue the work. The initial working prototype (system) should satisfy Amtech's management's demands and consider the most relevant set of features to guarantee excellent carpooling service.

The requirements Maarten has identified so far are:

1. The system should provide daily financial reports to management.
2. The system should provide weekly reports to CR managers regarding the overall customer perception of the service.
3. Users (drivers and passengers) should register in the system through the mobile application.
4. Users (drivers and passengers) should log in and be able to share content on social media.
5. Drivers should publish their trip on the application to find passengers to share the ride with.
6. The mobile application should support Android and iOS users.
7. The service should work smoothly and guarantee a 99.99% up-time.
8. Users should authorize sharing content on social media, and this should not be done automatically.
9. Passengers should be able to join the driver's trip and agree to all conditions specified (i.e. price).

10. Users should be able to modify their profile information.
11. The system should ensure the privacy of its users. Trips, social media accounts, and profile information are sensitive data.
12. Both the driver and passenger should be able to rate each other to gain reputation.
13. The system should offer a rapid response time for both drivers and passengers.
14. Drivers should be able to create a new trip to be displayed when passengers search for trips.
15. Passengers should be able to find a driver for a destination, when one exists. When passengers find a suitable trip, they can make a reservation, and a notification is sent to the driver.
16. Passengers should be allowed to specify their travelling preferences.
17. When either drivers or passengers arrive at the meeting point at the time agreed upon, they should check-in to notify the other user of their arrival.
18. Drivers should be able to register a frequent trip, indicating the origin and destination, departure and return times, as well as the frequency (daily or weekly).
19. Passengers should be able to search for a frequent trip that they can join. Passengers should specify the departing neighbourhood, destination, departure times, and frequency.
20. Only authorized users should be allowed to post and edit their information.
21. The system should match passengers' trip preferences with the best trip created by a driver.
22. Payments for the trips go through the mobile application.
23. The system should allow passengers to pick the meeting points in a map through the mobile application.
24. The system should support a high increment in the number of users appropriately. For instance, from 10 000 users to 100 000 users.

QUESTION: Please answer the following questions:

a. Having examined the initial set of requirements mentioned in the scenario, you now have the opportunity to begin your investigation.

   In the case of **Amtech's management (CEO, CTO, and CRM team)**, What suitable techniques would you suggest for eliciting and exploring their requirements?
   In each case, justify your choice of elicitation technique.

b. For this section, **examine ONLY the first eight (1 to 8) requirements** (proposed by Maarten) and identify which of them (or which parts of them) are Functional (FR) and which are Non-Functional (NFR). For the NFRs identify to which category they belong.

   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.

Answer 1-a.
We should use an interview with the CEO to understand the objectives of the project and to discuss why she wants the system to be scalable and how profitable she wants the application to be.

We can use an interview with the CTO to discuss her perspective on the project and I would better assert what data they want to be stored for both drivers and passengers.

We should use a workshop with the CRM team to brainstorm ideas on how to collect valuable feedback from the clients.

In the end, we should use a final workshop with the Management Team to discuss and record all the final requirements, to be sure that they all agree or come to a common point of view.

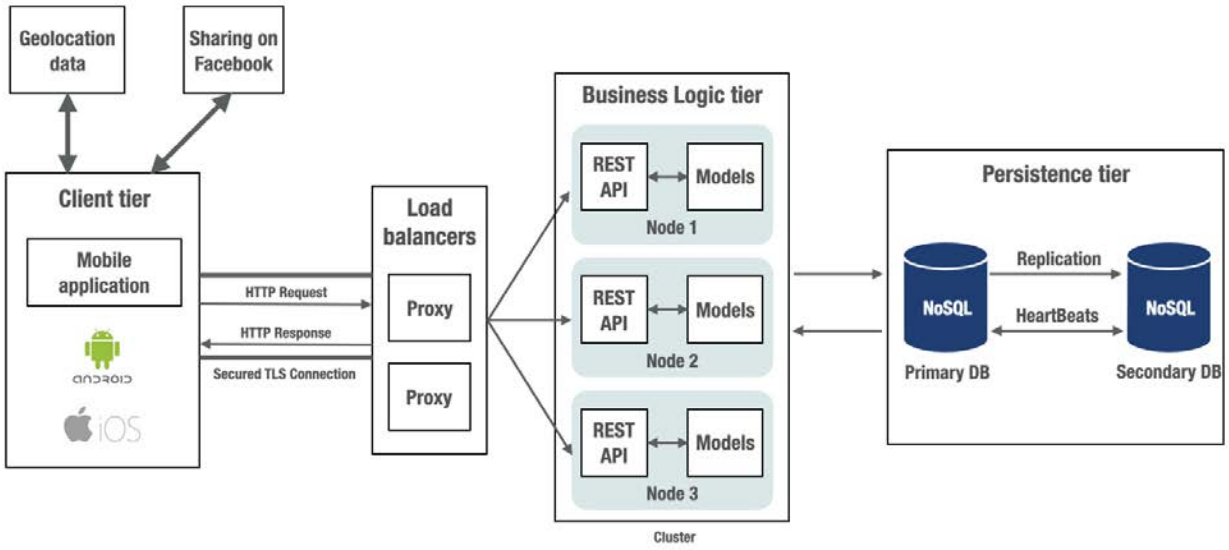
Answer 1-b.

- "The system should provide daily financial reports to management." is an FR.
- "The system should provide weekly reports to CR managers regarding the overall customer perception of the service" is an FR. "weekly"
    - Here we could benefit from a more detailed requirement on what the application should measure in terms of customer perception.
- "Users (drivers and passengers) should register in the system through the mobile application." is an FR
- "Users (drivers and passengers) should log in and be able to share content on social media." is an FR
- "Drivers should publish their trip on the application to find passengers to share the ride with." is an FR.
- "The mobile application should support Android and iOS users." is an NF-organizational requirement
- "The service should work smoothly and guarantee a 99.99% up-time." is

an NFR.
    - Usability and availability. Both are ambiguous, we don't know what smoothly means and we need a more detailed description.
- "Users should authorize sharing content on social media, and this should not be done automatically." is an NFR., External requirement.

2. For this question, consider the prior scenario in Question 1 (Carpooling Management System) and the extension below:

Figure 1: High-Level System Architecture



After several discussions between you, Maarten, and the startup team, the project has been divided into two stages. In the first stage, the focus is on developing the infrastructure to support the users' requirements (drivers and passengers). The second stage will include the features requested by Amtech's management. However, the initial architecture design should also support management requirements to avoid critical migration during the second stage.

For the first stage, the startup has proposed the architecture depicted in *Figure 1*. Bold arrows denote a secured connection using TLS (Transport Layer Security). Other connections represented by thin arrows occur in the Local Area Network (LAN), except for the HTTP request in a TLS tunnel.

The architecture proposed is a variant of the 3-Tier pattern. The application architecture has been chosen according to the previously specified requirements. The backbone of the application consists of a cluster of servers. A proxy has the responsibility of distributing the workload over the servers in the cluster. The cluster configuration allows adding as many servers as needed. A second load balancer (proxy) has been included in case one of them is down. In the Persistence tier, a primary database receives requests for retrieving and storing data. The architecture considers a replication strategy as well. For that purpose, the design includes a secondary database. Both databases exchange heartbeats for letting each other know its availability. In case of any incident with the primary database, the second database takes over the operations. A data redundancy strategy has also been taken into consideration to avoid data loss.
The business logic tier exposes a RESTful API that can be consumed by any REST client. The data exchanged will be formatted using JSON.
The client tier supports Android and iOS. The Android/iOS mobile application communicates with the server using TLS.
Finally, The system interacts with Facebook (Sharing service) by using Facebook's APIs (web services).

QUESTION: Please answer the following questions:

a. Were quality attributes such as ***Performance*** and ***Scalability*** taken into account when choosing the architectural pattern? Why? (Motivate your answer describing explicitly your rationale and state your assumptions clearly).

b. Do you agree with the initial architectural design proposed by the startup? Why? What would you have changed in the architecture? (Motivate your answer considering **ONLY the quality attributes mentioned in (a)**, present your arguments and state your assumptions clearly).

Answer 2-a.

Performance - Although in general, the layered approach can mean requests are made all through the same chain all the time, here certain responsibilities have been taken to avoid slowness: the client directly connects to the external APIs (no need to request them from the proxy + business logic tier, unnecessary overhead). The proxies also ensure they won't be times of random or unexpected slowness or going down. This would be a lot more frustrating for users, especially those that have important destinations to reach

a deadline.

Regarding scalability, we take it into account when services can run on different servers within an environment. Thus, the system will manage the situation when it will have to increase the load quickly, without losing the performance. However, if scalability is the main quality attribute to consider, there are more scalable architectures that could have been considered. For example, the SOA (Service - Oriented Architecture) will have higher scalability since services can run on different servers within an environment, or in Publish-Subscriber Pattern we will also have higher scalability since we can have as many publishers or subscribers we want.

Answer 2-b.

A better choice is to use the SOA because in this kind of scenario we will care about the performance and scalability. On the other hand, we can assume that in the future new features will be added; hence, an SOA version is very suitable for it.

Moreover, because using SOA we can scale everything very easy and we can use as messages JSON format between the services and as a registry, we will have a list of all available services, used by the service to discover other services to use. Furthermore, the system will not have a worse performance by using this architectural pattern.

3. Consider the following scenario:
   "*We want to implement an online system for a large international bank. The system should allow users (bank customers) to create and monitor their investments. Users should be able to create multiple types of investments with different markets (e.g., European market, US market, etc.)*
   Each market has a portfolio of investment types:
   - *Saving accounts*
   - *Bonds*
   - *Stocks*
   - *Insurances.*

   Each investment has an interest rate, risk level, and duration. These investment attributes vary across investment types (e.g., savings accounts have lower interest rates than bonds) and across the market.

For example, if the user wants to invest in bonds in the European market, then the system should create European bonds with its specific attributes (low-interest rate, low risk level, short duration). If the user wants to invest in stocks in the US market, the system should create US stocks with their specific attributes.

We want to design the system in such a way that it allows us to model all possible combinations between markets and the types of investments with minimum code duplication. We also want to make the system open for future extensions (i.e., adding more markets and more types of investments)."

Part 1: Which design pattern fits best in the implementation of this system? Please explain your choice.

Part 2: For the given design pattern of your choice (part 1), indicate

- 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.

Answer 3-a.
The most suitable pattern is Abstract Factory because a system should be independent of how its products are created, composed, and represented. Basically, we will have a factory for investments and we will have a factory for the markets. Moreover, we want to provide a class library of products, and I want to reveal just their interfaces, not their implementations. Thus, multiple models can be added in the future.

Answer 3-b.
Component: Investment
- Role: the Client
- Methods: createMarket(), createPortofolio()

Component: Market
- Role: Abstract product
- Methods: createMarket(), createPortofolio()

EUMarket
- Role: Concrete Product
- Methods: createProduct()
- Association: It extends (inherits) the classMarket

UsMarket
- Role: Concrete Product
- Methods: createProduct()
- Association: It extends (inherits) the classMarket

Portofolio
- Role: the AbstractFactory
- Methods: createMarket(), createProtofolio()

- Role: Concrete Product
- Methods: createProduct()
- Association: It extends (inherits) the classProtofoliou

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

   IMPORTANT: To avoid fraud and student cooperation during the exam, you have to adapt the coverage matrix based on your student number. Complete the coverage matrix based on the instructions below:

   Mark with "x" the entries in the grey area of the table whose test case index corresponds to the last three digits of your student number. For example, if your student number is XXX521, you have to mark with "x" the entries [t5, b1], [t2, b1], and [t1, b1].

   If one of the last three digits of your student number is zero you ignore that digit. For example, if your student number is XXX520, you have to mark with "x" the entries [t5, b1], and [t2, b1].

   If the last three digits of your student number contain duplicates, you ignore the duplicates. For example, if your student number is XXX522, you have to mark with "x" the entries [t5, b1], and [t2, b1].

   QUESTION: Please answer the following questions:

   a. Which test cases are selected when using the additional greedy algorithm to reach 100% branch coverage?
   b. Which test cases are selected when using the greedy algorithm to reach 100% branch coverage?

Report the test case(s) in the exact order they are selected by the additional greedy algorithm (point a) and the greedy algorithm (point b).

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	x												
t5									x	x				x					x	x
t6		x	x									x	x							
t7															x	x	x			x
t8	x		x	x	x			x	x										x	x
t9			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

Answer 4-a.
t8, t3, t6, t7, t4, t10

Answer 4-b.
T8, t4, t1, t2, t3, t10, t12, t14, t5, t9, t11, t15

5. Consider the program in Listing 1 and its mutant in Listing 2. Note that the mutation has been applied to line 46.

IMPORTANT: To complete the program, fill in the constant A, B, and C (lines 3-5) following the instructions below:

From the list {2, 3, 4, 5, 6, 7, 8, 9} select the three digits from left to right that are not in the last three digits of your student number. For example, if your student number is XXX521, you have to select the constant A=3, B=4, C=6. This additional complexity has been introduced to avoid fraud and student cooperation during the exam.

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.

In your answer, also specify the values of three constants A, B, and C that you have used.

Listing 1

```
1.  public class CoinChanger {
2.     int A =      // To be filled-in
3.     int B =      // To be filled-in
4.     int C =      // To be filled-in
5.
6.     private Integer[] coins = {1, A, B, C};
7.
8.     /**
9.      * This method computes the minimum number of coins that add up to
10.     * a given amount of money.
11.     * The change-making problem is solved via recursion
12.     * @param amount the amount to reach with the existing coins
13.     */
14.    private int change(int amount) {
15.        int count = 0;
16.
17.        // sort the coins in ascending order
18.        Arrays.sort(coins);
19.
20.        // Traverse through all denomination
21.        for (int i = coins.length - 1; i >= 0; i--) {
22.            // Find denominations
23.            while (amount >= coins[i]) {
24.                amount -= coins[i];
25.                count++;
26.            }
27.        }
28.        return count;
29.    }
30.
```

```
31.     /**
32.      * This method computes the number of coins to return to reach a
33.      * certain amount. If the number of coins to give is above
34.      * {@code maxCoins}, this method will give -1 as results
35.      * to indicate that it is better use banknotes rather than coins
36.      * @param amount amount to change with coins
37.      * @param maxCoins max number of coins to use for change
38.      */
39      public int compute(int amount, int maxCoins) {
40.         if (maxCoins < 2 || amount <= 0) {
41.             // giving one coin is always acceptable
42.             throw new IllegalArgumentException();
43.         }
44.
45.         int change = this.change(amount);
46.         if (change > maxCoins){ // too many coins, let's use banknotes
47.             return -1;
48.         } else {
49.             return change;
50.         }
51.     }
52. }
```

Listing 2

```
1.  public class CoinChanger {
2.     int A =      // To be filled-in
3.     int B =      // To be filled-in
4.     int C =      // To be filled-in
5.
6.     private Integer[] coins = {1, A, B, C};
7.
8.     /**
9.      * This method computes the minimum number of coins that add up to
10.     * a given amount of money.
11.     * The change-making problem is solved via recursion
12.     * @param amount the amount to reach with the existing coins
13.     */
14.    private int change(int amount) {
15.        int count = 0;
16.
17.        // sort the coins in ascending order
18.        Arrays.sort(coins);
19.
20.        // Traverse through all denomination
21.        for (int i = coins.length - 1; i >= 0; i--) {
22.            // Find denominations
23.            while (amount >= coins[i]) {
24.                amount -= coins[i];
25.                count++;
26.            }
27.        }
```

```
28.         return count;
29.     }
30.
31.     /**
32.      * This method computes the number of coins to return to reach a
33.      * certain amount. If the number of coins to give is above
34.      * {@code maxCoins}, this method will give -1 as results
35.      * to indicate that it is better use banknotes rather than coins
36.      * @param amount amount to change with coins
37.      * @param maxCoins max number of coins to use for change
38.      */
39      public int compute(int amount, int maxCoins) {
40.         if (maxCoins < 2 || amount <= 0) {
41.             // giving one coin is always acceptable
42.             throw new IllegalArgumentException();
43.         }
44.
45.         int change = this.change(amount);
46.         if (change >= maxCoins){ // <--- MUTATED LINE
47.             return -1;
48.         } else {
49.             return change;
50.         }
51.     }
52. }
```

Answer 5.

```
// A=2
// B=4
// C=5
@Test
public void test() {
    CoinChanger cc = new CoinChanger();
    assertEquals(3, cc.compute( 12,3));
    // the change will be equal to maxCoins,
    // this will kill the mutant
}
```

6. Consider the program in Listing 3. Answer the following questions:
    a. How many edges are in the Control Flow Graph of the `countPrimes(...)` method?
    b. How many nodes are in the Control Flow Graph of the `countPrimes(...)` method?
    c. What is the Cyclomatic Complexity (CC) of the `countPrimes(...)` method?

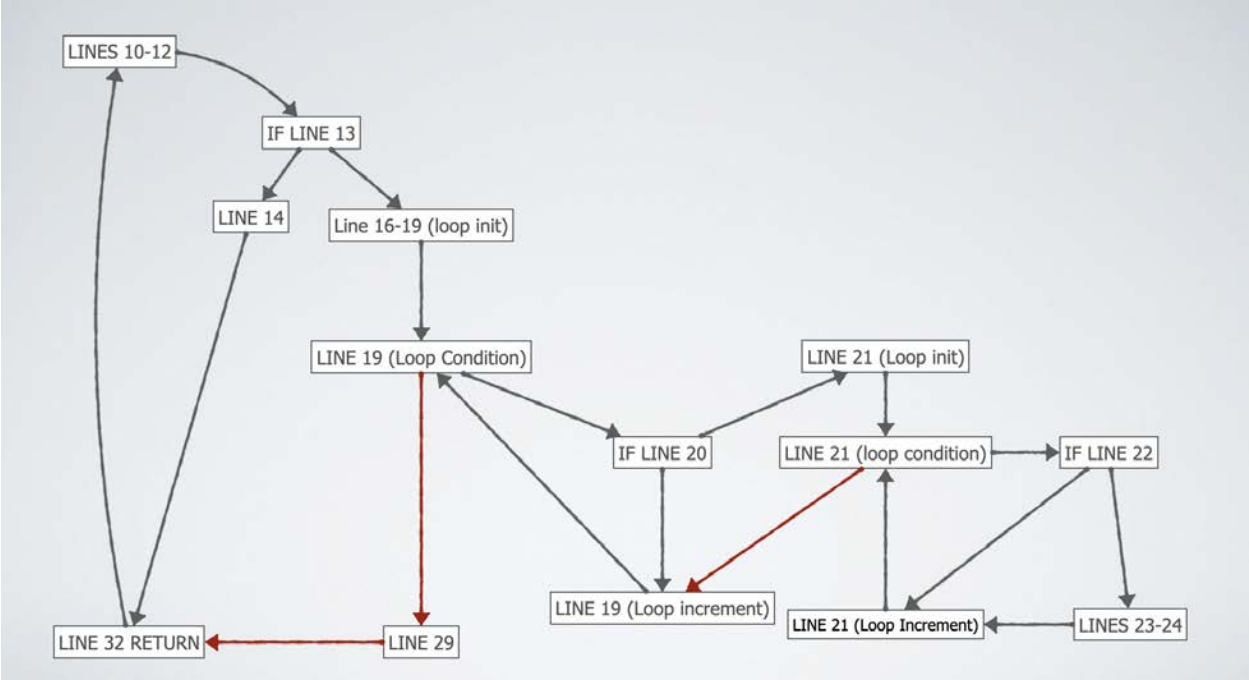
Listing 3

```
1.  public class PrimeNumber {
2.
3.      /**
4.       * This method computes the number of prime numbers smaller than
5.       * or equal to a given positive number
6.       * @param number upper bound
7.       * @return number of prime numbers <= {@code number}
8.       */
9.      public static int countPrimes(int number)  {
10.         int results = 0;
11.
12.         if (number > 1) {
13.             if (number < 4)
14.                 results = number - 1;
15.             else {
16.                 BitSet set = new BitSet();
17.                 int s = (int) Math.sqrt(number);
18.                 int counter = number;
19.                 for (int p = 2; p <= s; p++) {
20.                     if (!set.get(p)) {
21.                         for (int q = 2; (p * q) <= number; q++) {
22.                             if (!set.get(p * q)) {
23.                                 counter--;
24.                                 set.set(p * q);
25.                             }
26.                         }
27.                     }
28.                 }
29.                 results = counter - 1;
30.             }
31.         }
32.         return results;
33.     }
34. }
```

Answer 6-a.
Number of edges = 19

Answer 6-b.
Number of nodes = 14

Answer 6-b.
CC = 19 - 14 + 2 =7


LINES 10-12
IF LINE 13
LINE 14
Line 16-19 (loop init)
LINE 19 (Loop Condition)
LINE 21 (Loop init)
IF LINE 20
LINE 21 (loop condition)
IF LINE 22
LINE 19 (Loop increment)
LINE 21 (Loop Increment)
LINES 23-24
LINE 32 RETURN
LINE 29