Linköpings Universitet
IDA
Kristian Sandahl

2020-10-27

Written exam for Software Engineering Theory

Course codes TDDC88, TDDC93, 725G64

Note: When we visit the exam, we will take a slow walk among all students, so you don't need to sit with your hand raised. Just call for our attention when we pass your desk.

Instructions to students, please read carefully

- **Explicitly forbidden aids:** Textbooks, machine-written pages, photocopied pages, pages of different format than A4, electronic equipment.
- Try to solve as many problems as possible.
- Motivate all solutions.
- Please, write and draw clearly.
- Write solutions for different areas (fundamental part) and different problems (advanced part) on separate sheets of paper.
- Label all papers with AID-number, date of examination, course code, examination code, and page number.
- You may write solutions in either Swedish or English.
- Please, note that the problems are not necessarily written in order of difficulty.
- TIP! Read all exercises in the beginning of the exam. This will give you the possibility to ask questions about all parts of the exam since the examiner will visit you in the beginning of the exam time.

Grading

The exam consists of two parts: Fundamental and Advanced.

The Fundamental part has problems worth 10 credits per area. Areas are Requirements, Design & Architecture, Testing & SCM, Planning & Processes, and Software Quality. Thus, the Fundamental part can give maximally 50 credits.

The Advanced part has problems worth 50 credits in total. Each problem typically requires a solution of several pages.

The maximum number of credits assigned to each problem is given within parentheses at the end of the last paragraph of the problem.

Pass condition: At least 4 credits per area in the Fundamental part **and** at least 50 credits in total. The total amount of credits also includes the bonus credits you might have got in lecture exercises autumn 2020. This gives you the mark 3. If you have at least 4 credits for 4 of the areas in the Fundamental part, then you can still pass if you have more than 60 credits in total.

Higher marks are given based on fulfilled *pass condition* **and** higher amounts of credits according to the following table:

Total credits	Mark
0-49	U (no pass)
50-66	3
67-83	4
84-	5

Good Luck!

Kristian

Problems

Part 1: Fundamental

Area 1: Requirements

1 a) Requirements in a good specification shall be *numbered* and *inspected.* Explain why this is good. (2)

1 b) Write down two *use-cases* with two different *actors* of the Zoom system that we used in the off-site lectures. Also draw a *UML use-case diagram*.
Remember to include the *use case* textual descriptions. Use full sentences. Logging in to the Zoom system is not a *use-case* of its own since involves so few interactions. (4)

1 c) Write down two *functional* and two *non-functional* requirements of the Zoom system that we used in the off-site lectures. (4)

Area 2: Design and Architecture

2 a) Explain two ways in which you can use *prototyping* in the design of a software system. (2)

2 b) *Scenario:* A system for simulation of an aircraft is divided into several smaller simulators. Even though they are small they require extensive amount of CPU power to do their calculations. To federate the results there is a layer monitoring the tasks for each simulator. The layer also integrates the result from the simulators to form the system level simulation results. Even though the task of this layer is simple in principle, it contains quite much numerical processing. The end-user is a programmer who wants the result to check that his/her software works with the rest of the aircraft. That person is using a standard PC.

Task: Which type of *client-server architecture* matches the scenario best? Explain why? For the chosen *architecture*, mention two advantages or two drawbacks. One advantage and one drawback also comprise a good answer. (4)

2 c) *Scenario:* In a checkout system in a shop the customer can pay with a debit card, a credit card, or Swish[1]. If the customer pays with card, the card reader is invoked by the registration unit, who also sends the sum to pay to the card reader. The customer inserts the card, approves the sum, enters the code, gets an OK from the card reader, and removes the card. If the customer pays with Swish, the screen of the checkout terminal is invoked which displays a QR-code for the payment. The customer opens the Swish app in his/her mobile unit and scans the QR-code. The Swish app presents the sum and recipient. When user approves, the mobile BankID app is opened where the sum and recipient is shown again. To complete the transaction, the user enters the password, and gets an acknowledgement. Regardless of payment method, the customer gets a receipt.

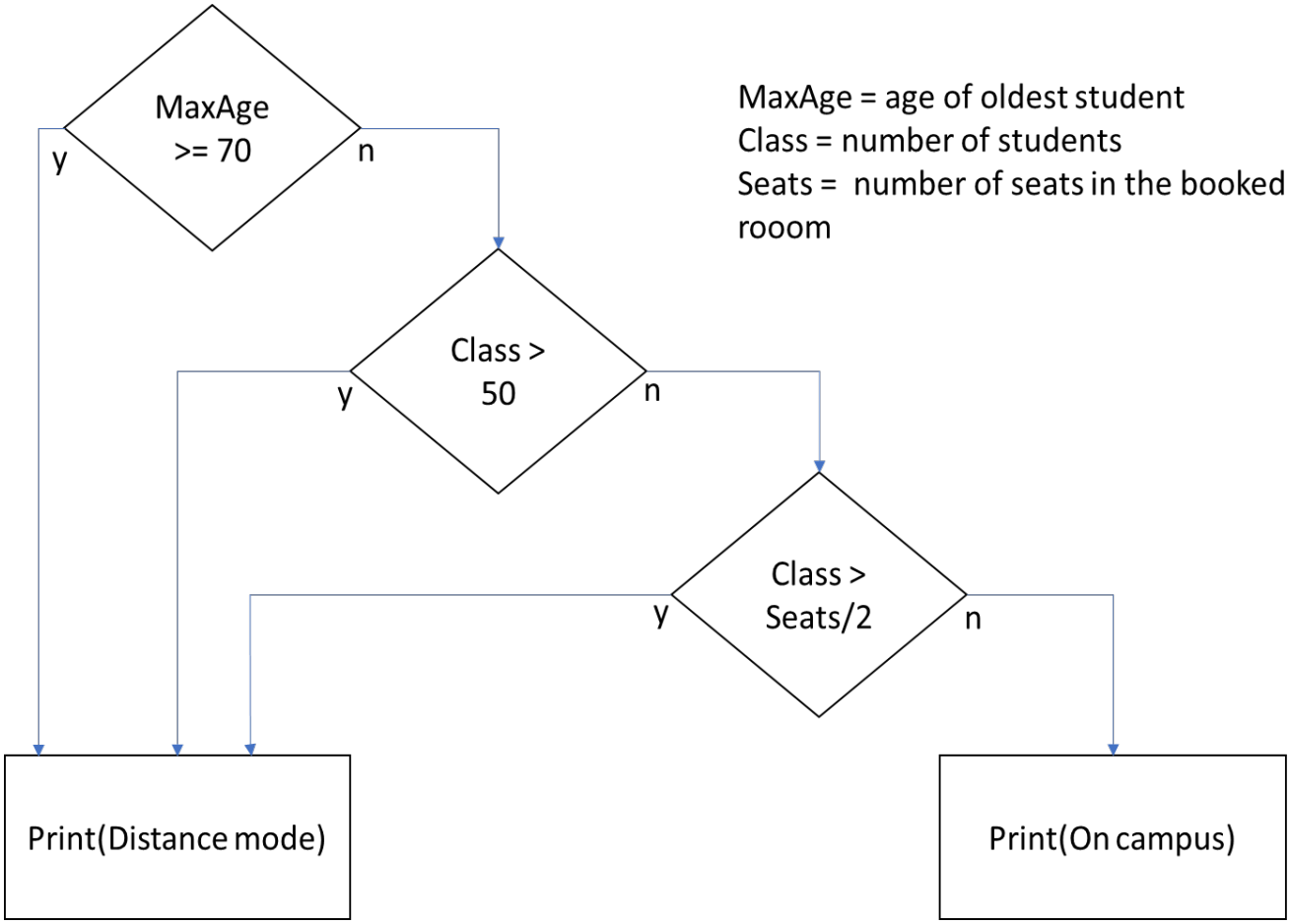
Task: Model the interactions between the customer and the equipment in the scenario with a *UML Sequence diagram*. Use at least one *fragment*. You can assume that there is a single bank server taking care of authentication and transactions between the customer and the shop. (4)

[1] Swish is a mobile app that is used for payments between private persons in Sweden, but also from private persons to companies and non-profit organizations. It has many similarities with apps for PayPal.

Area 3: Testing and SCM

3 a) What is the difference between a *fault* and a *failure* as defined in this course? (2)

3 b) Study the following *flowchart* for an education mode advisor for a lecture during the Covid-19 epidemic.



Create a *test table* for *test cases* guaranteeing *full path coverage* with the minimal number of *test cases*. Is there a difference in the minimal number of *test cases* if we were satisfied with *statement coverage*? Remember to motivate your answer! (4)

3 c) Describe the practice of *Continuous Integration*. Also describe two advantages of the practice. (Hint: We use the definition given at the lecture on *Continuous practices*, that is, no *build* and *test* are necessarily involved.) (4)

Area 4: Planning and Processes

4 a) What is the difference between a *milestone* and a *tollgate*? (2)

4 b) Describe four different *plans* for handling the *risk* "It might be hard to convince a large enough number of end-users to participate in the usability evaluations necessary for our prototyping approach." (Hint: If you cannot come up with *plans* for the *risk*, suggest your own *risk*.) (4)

4 c) The *classical waterfall life-cycle model* consists of several distinct *phases*, such as *requirements*, *architecture*, *design*, etc. Describe the principles of the *classical waterfall life-cycle model*. Focus on the principles, not the *phases*. Write also down two advantages with the *classical waterfall model*. (4)

Area 5: Software Quality

5 a) Explain why the formula $MTTF/(MTTF + 1)$ can approximate *reliability*. (Hint: MTTF = Mean Time To Failure.)

5 b) Describe four different *roles* involved in an *inspection*. (4)

5 c) Scenario*:* You have developed a very good prediction algorithm and have lot of customers who like you to provide them with applications around it. That is good, but internally the work is chaotic, all people make small contributions to almost all projects prioritizing work according to the earliest-deadline-first principle. No one seems to have an overview of which state the different projects are in. The customer satisfaction data has recently been showing a large variance.

Task: Describe a *CMMI process area* that you think will help you the most. A description is typically 5-6 sentences; that covers *Introductory notes* and/or *Specific goals*. Also, describe how this will help your company. (4)

Part 2: Advanced

6. Describe 2 different *metrics* for *usability* of a software system. For each metric we want to know:

- Description
- How to obtain data
- How to calculate the *metric*
- Thorough argumentation for why the *metric* can indicate low/high *usability*. (Hint: 3-4 sentences)
- Which type of *prototype* the metric can be used for (*Lo-Fi* or *High-Fi*)? Remember to motivate your answer. (10)

7. There are several *quality factors* that can be attained by the design of an *architecture* and/or the *way of working (process* or project organization).

Select five *quality factors* and for each factor, explain how it can be dealt with by either the *architecture* or the *way of working*.

For example, the quality factor performance can be improved by selecting a parallel processing architecture. Parallel computing divides the computation in smaller steps that are executed simultaneously on different processors. Thus, we get the final result in a shorter time interval compared to a single processor execution. This gives a net improvement of time, even if we have to count with some overhead in dividing the task and integrate the solutions. (end example)

To help you thinking we append below a list of quality factors from the lecture:

- Correctness
- Reliability
- Efficiency
- Usability
- Integrity
- Maintainability
- Flexibility
- Testability
- Security
- Portability
- Reusability
- Interoperability
- Survivability
- Safety
- Manageability
- Supportability
- Replaceability
- Functionality
- Business quality

You may formulate other *quality factors* (10)

8. *Scenario:* It's good to take a break from the studies by going to a student event now and then. However, it is wasted time to stand in the queue for tickets in 14 hours, when you could sleep or study software engineering instead. Therefore, all LiU event organizers will have a virtual queue system where you place your bot instead of yourself. The idea is that you place the bot in a queue. From the queue the bot can be invited to the lobby. If your bot is invited into the lobby it is guaranteed to purchase a ticket for you. Invitation to the lobby from the queue is basically random, but the bot has an advantage if it has earned queue credits earlier.

The following features are possible

- In the queue your bot can look for your friends' bots. If the bots agree they can queue as a group of 2, 3, or 4 bots.
- The information for grouping can be set by yourself and/or be inferred from social media. The weight of the information sources can be set in advance.
- At a certain time, the virtual queue randomly selects individual bots and groups to the lobby.
- The algorithm mixes individual bots and groups in a way that makes the probability for an individual bot to make it for the lobby fairly equal regardless of if it is queueing as a group member or as an individual.
- The algorithm can be seeded to give preference to bots with earned queue credits.
- If your bot is not selected for the lobby, your bot earns 5 queue credits.
- In the lobby the bots are asked from the system to determine preferences for the event, for instance, dietary restrictions, what to drink etc.
- In the lobby the bots discuss and build swarms of bots whose owners want to sit close to each other. A swarm consists of an individual bot or groups of bots.
- The bots in the swarms discuss what you should wear and decides on where and when to meet before the event.
- If a bot feels lonely and wants to leave the lobby, it can do so and earns 20 queue credits. A new bot is then invited to the lobby.
- At a certain time, all tickets are distributed to all bots in the lobby according to their preferences.
- The bot sends you an alarm to pay for the tickets within 30 minutes, otherwise you are out, and a new person's bot is invited.
- The bot moves into your mobile device and joins the event to collect data, for instance how much you are dancing. After the event, the bot asks you some questions to adjust your preferences.

Task: Draw a *UML Class diagram* of all concepts described above, use at least one *generalisation* and one *composition* relation. Write name and *multiplicity* on the *associations*. (Hint: Some concepts can be better modelled as an *attribute* of a class rather than a class of its own. You may introduce new concepts or features if you explain that in the beginning of the solution.) (20)

9. *Scenario:* Same as in problem 8.
Task: Create a *UML State diagram* of the *class* Bot. (10)