Linköpings Universitet
IDA
Kristian Sandahl

2019-01-08

Written exam for Software Engineering Theory

Course codes TDDC88, TDDC93, 725G64

Note: When I visit the exam, I will take a slow walk among all students, so you don't need to sit with your hand raised. Just call for my attention when I 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, Planning & Processes, Design & Architecture, Testing & SCM, 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 longer 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 2018. 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

Multiple choice questions

In multiple-choice questions, we will ask you to write down the letters A, B, C, or D for the one or two statements that you think are true. Note that you should not write down the statements that you think are false. There are exactly two true statements per question, so answering with three or four alternatives with gives 0 credits.

For each statement that you select that is correct (i.e., that the statement is in fact true) you get one credit. For each statement that you select that is incorrect (i.e., that the statement is in fact false, but you believed it was true) you get minus one credit. Each multiple-choice question can give maximum 2 credits and minimum 0 credits, i.e., you cannot get negative credits for one multiple choice question.

Example 1: Assume that you have written down statements A and C. If now statements A and B were true, and statements C and D were false, you would get +1 credit for writing down A, but -1 credit for writing down C. Hence, the total credits for the multiple-choice question is 0.

Example 2: Assume that you have written down statement B. If now statement A and B were true, and statement and statement C and D were false, you would get +1 credit for the multiple-choice question.

Example 3: Assume you correctly wrote both statement A and B. If now statements A and B were true, and statements C and D were false, you would get +1 credit for writing down A, and +1 for writing down B. Hence, the total credits for the multiple-choice question is 2.

Good Luck!

Kristian

Problems

Part 1: Fundamental

Area 1: Requirements

1 a) Which of the following statements are true? Answer with the statement letter only, no motivation is needed. (2)

A. *Requirements elicitation* is the process of formulating known requirements in a mathematical notation.
B. Two requirements that do not contradict each other are said to be *traceable.*
C. *Entity-relationship modelling* is a kind of *conceptual modelling.*
D. *Unambiguous* requirements can only be interpreted in the intended way by the developers.

1 b) *Scenario:* A modern library system contains catalogues of books and journals. It can also give the users access to on-line information databases. In addition to this, the library provides citation-handling system for users writing reports. There is a member management system recording data about the members including volumes that are on loan or reservations made. Members can also be suspended if they don't follow the rules or damage library property. The librarians can also use external services, such as, other libraries or book vendors.

Task: Create a *use-case diagram* for a new library system consisting of two different actors and two different use-cases. Don't forget the use case description. Logging in and logging out are basic functions, not to be considered as use-cases. (4)

1 c) Describe the concept of user stories. You may use an example. Describe two situations when user stories is a good choice of documenting requirements (4)

Area 2: Planning and Processes

2 a) Which of the following statements are true? Answer with the statement letter only, no motivation is needed. (2)

A. The *Waterfall* life-cycle model can be a good alternative for fixed-priced contracts.
B. An *agile method* constantly adapts the requirements to fit the system being developed.
C. The definition of an *iterative* life-cycle model is that components are developed and integrated with a fully working sub-system.
D. A potential drawback with an *iterative* life-cycle model is that extra administrative overhead for planning and coordination is added.

2b) Suppose you are developing software in *time-boxed iterations*. Name each of the four *dependent project parameters* and describe if it is locked or free to change. Give two examples of what you can do if one of the parameters changes. (4)

2c) Describe two ways of estimating the effort needed for a software development project. (4)

Area 3: Design and Architecture

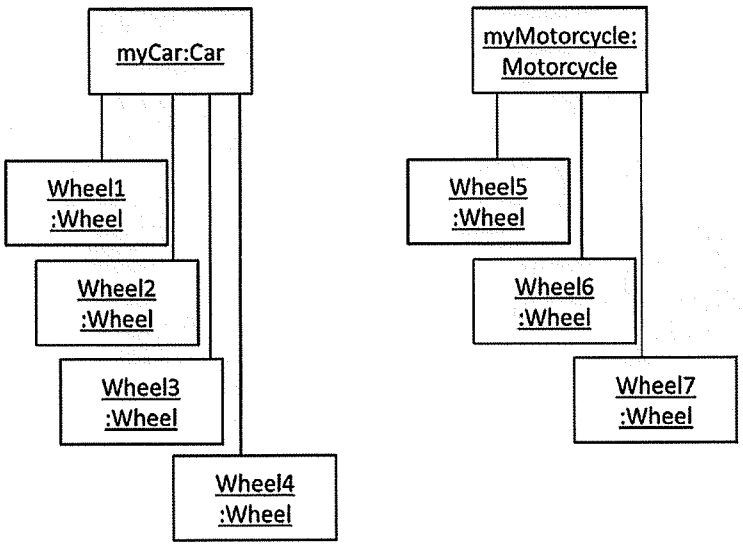
3 a) Which of the following statements are true? Answer with the statement letter only, no motivation is needed. (2)

A. A *three-tier client-server architecture* supports the use of intelligent load balancing
B. A *layered* architecture supports *incremental* development and testing
C. *Layer bridging* is a concept of *pipe-and-filter* architectures that means that some data can take a short-cut by passing a series of *filters* in a *pipe*.
D. In a *two-tier, fat-client architecture* the *presentation layer* is on the server-side, which sends ready-made displays to the *clients*.

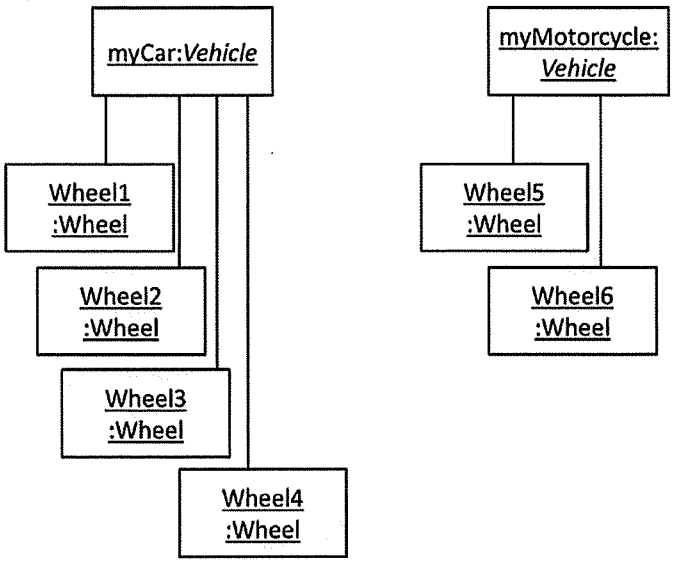
3 b) Name two different *design patterns*. For each of the patterns, describe what problems the particular pattern can solve. (4).

3 c) Draw a *class diagram* of the classes Vehicle, Car, Motorcycle, and Wheel that allows the instantiations in I:, but not the ones in II:, and not those in III: (4).

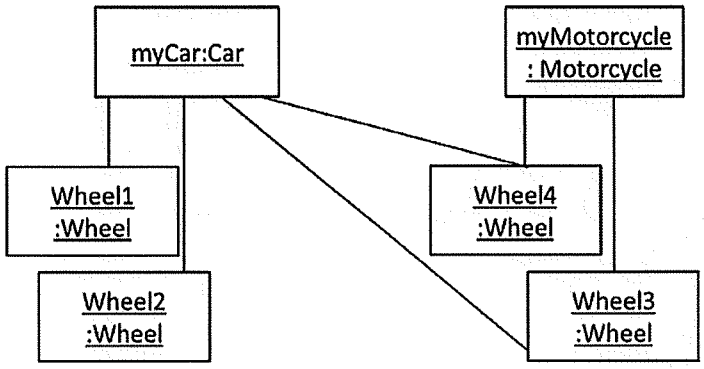
I:



II:


myCar:Vehicle
myMotorcycle:
Vehicle
Wheel1
:Wheel
Wheel5
:Wheel
Wheel2
:Wheel
Wheel6
:Wheel
Wheel3
:Wheel
Wheel4
:Wheel

III:


myCar:Car
myMotorcycle
: Motorcycle
Wheel1
:Wheel
Wheel4
:Wheel
Wheel2
:Wheel
Wheel3
:Wheel

Area 4: Testing and SCM

4 a) Which of the following statements are true? Answer with the statement letter only, no motivation is needed. (2)

A. A *baseline* is a configuration of the *base classes* in an object-oriented program.
B. A *development branch* is a sub-committee of the *change control board.*
C. When two persons do a *check-out* in a *centralized modify-merge* version handling system they both create their own copy of the file.
D. A *configuration item* can be other important artifacts, not just source code.

4 b) *Scenario:* You are developing a system checking whether goods can be sent as a Swedish domestic letter. The rules are:

Minimal sizes

Cuboid: Width = 90 and length =140 mm

Cylinder: Length: 100 mm, length + 2 x diameter: 170 mm

Maximal sizes

Cuboid: Length: 600 mm
Cuboid: Length + width + thickness: 900 mm

Cylinder: length 900 mm, length + 2 x diameter: 1040 mm

Task: Your task is to identify at least 4 *equivalence classes* and provide one *test case* per equivalence class for your software. You may assume that you can detect whether goods is of either a cuboid or cylinder. (4)

4c) Describe the concepts from system level testing: *Function test, performance test, acceptance test,* and *installation test.* (4)

Area 5: Software Quality

5 a) Which of the following statements are true? Answer with the statement letter only, no motivation is needed. (2)

A. The *author* participating in an *inspection* meeting must only listen even if he or she detects defects.
B. An *audit* is a *software review* with no written documents; the participants discuss about the ideas of the software.
C. You normally record data about the inspection itself in order to improve the expensive inspection process.
D. Both the *author* and the *inspection leader* take part in the *exit-and-follow-up phase* of the inspection process.

5 b) Suggest two different *software metrics* for measuring the *maintainability* of software. Don't forget to motivate why the metrics can give information of the maintainability. (4)

5 c) *Scenario:* Life isn't easy: Your end-users complain that even though your software is failure-free and has a good looking GUI, you miss certain functions that "everyone" in their company know are sometimes needed. Your employees are really appreciating your personal feedback, but it comes sporadically and is based on your feelings only. As a complement, they would like to have more regular and objective feedback.

Task: Your task is now to find the two most relevant *CMMI process areas* that can help you to take care of the criticism. Write down their names and a short motivation. (4)

Part 2: Advanced

6. *Scenario:* Suppose that you've got the job to develop a computerized examination system for Linköping University. The examinations are taken by students in a specially equipped room as soon as they feel ready. Biological sensors are used to determine the identity of the students, and electric sensors are used to determine that no communication equipment is brought into the room. Most problems are graded automatically. There is grading support for essay questions using topic and phrase extraction from answers. The requirements on security and reliability are high.

You and your four team members start the development project 9 January 2019 and start *alpha testing* 30 August 2019. You need to use formal procurement procedures to buy the hardware. This includes publishing specifications, await three weeks for bids, evaluating bids formally, publish the evaluation result, await the appeal period, and write a contract with the chosen vendor.

Since this is an EU-project you get payed in EUR and need to outsource some of the development to another EU country.

Task: Make a list of five relevant *risks* that need to be monitored. Use your risks to demonstrate how you can calculate the *risk magnitude indicator*. Also, make a *plan* for each of the risks. Select your risks so that you can give examples of four different types of risk planning: *Risk avoidance, risk transfer, risk mitigation,* and *definition of a contingency plan*.
Hint: You don't have to give examples of all types of risk planning for each of the risks. It is sufficient if all types of risk planning occur in your entire solution. So risk no 1 can have risk avoidance, risk no 2 risk has risk transfer, etc.
(10)

7. Make a comparative analysis of the three agile frameworks. *SCRUM*, *Kanban*, and *eXtreme Programming* (XP). For each of the frameworks write:

- A short description of what developers and end-users can expect from the framework. 2-3 sentences
- A list of important concepts of the framework, for instance, roles, artefacts, meetings, and technical practices. For each concept, make a 1-2 sentence description and give a short motivation if you think this concept is unique to the framework or if there is a similar concept in another framework.

For full credits you describe at least two concepts per framework and 7 concepts in total. (20)

8. *Scenario:* You are developing a global peace stabilization system (GPSS). The idea with the system is to collect lot of information about potential conflicts in the world. The more different evidence you have, the stronger is the degree of alert:

1. **Green**: the region is observed due to some incidents.
2. **Yellow**: there are conflicts in the region.
3. **Red**: immediate attention by the United Nations is required.

You have several sources classified as:

- **Indicators**, for instance, changes in the argumentation in media.
- **Reliable**, for instance, video clips taken by habitants.
- **Highly reliable**, for instance, satellite photos.

Each source has its own data collection method, data base, and completely automated data analysis. The output from each source is the percentage of conflict likelihood and a threshold value, indicating that conflicts are detected.

The rules for the alerts are:

- Green: two indicators pass threshold value, or a reliable source is 10 per cent units below its threshold.
- Yellow: two reliable sources pass the threshold, or one highly reliable source is 10 per cent units below threshold.
- Red: one highly reliable source passes the threshold and two reliable sources pass the threshold.
- If the conditions for the current alert level are changed, the level is lowered to the level where the conditions are fulfilled.

Task:

Show with an example *UML class diagram* how the *strategy design pattern* can be used to model the decision-making software, called Black-board, and its relation with the sources. Don't forget that the diagram probably needs a small explaining text. If you have forgotten the design pattern, you can get some credits for a general UML class diagram. (10)

9. *Scenario:* The same as in problem 8.

Task:

Make a *UML State chart* for a class named `AlertLevel` that holds the current level of alert. The rules specified above shall be obeyed. If you make assumptions about the implementation of the system, these must be written down. (10)

Hint: you need a no-alert state.