Linköpings Universitet
IDA
Kristian Sandahl

2017-08-24

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 through 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 2016. 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 statement A and B were true, and statement and 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. A requirement is said to be ambiguous if it cannot be interpreted in more than one way.
B. A requirement is said to be traceable if it is easy to find the software components that realize the requirements.
C. Requirement analysis involves conceptual modelling, for instance use-cases
D. Requirements validation is achieved when all requirements are correctly spelled and made up by complete sentences.

1 b) *Scenario:* You are designing a system for a gym where data about members and their progress is stored. The personal data of the members are used by both members themselves, administrators and the receptionist. The data comprises name, address, phone, membership category, membership validity, payment options, etc. The progress data is used by members and the personal trainers. Typically, progress data consists of types of exercises, workload, number or repetitions, training record, personal goals etc.

Task: Write at least two use-cases, with at least three different actors of the gym system introduced above. Draw a UML use-case diagram of the use-cases. (4)

1 c) Describe one good and one bad property of each of the following two requirements R1 and R2

R1: The processing time of an average customer registration shall be optimal.
R2: The customer shall be able to login with a Facebook-ID and then be able to listen to music from the personal playlist.

(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. Using the classical waterfall model will let you spend more time on testing compared to an iterative model.
B. The classical waterfall model has more similarities to the life cycle of hardware design than an iterative model.
C. Using an iterative model is the preferred model for fixed-price contracts.
D. When using an incremental model you build the system by implementing and testing parts of the system in small steps.

2b) Describe one similarity and one difference between SCRUM and Kanban. How is the difference motivated by proponents for SCRUM and Kanban respectively? (4)

2c) Briefly describe four items that you would typically include in the project organization part of a project plan. One sentence description and one sentence motivation per item is sufficient. (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. The decompose-and-compose design approach does not give any benefits if you do not intend to use an object-oriented language.
B. The decompose-and-compose design approach prescribes that you first attain coupling and then cohesion.
C. Low coupling of a design makes it easy to isolate a change or a fault.
D. High cohesion of a design makes it easy to locate where to perform a decided change.

3 b) Describe two benefits of using a layered architectural style. One sentence description and a motivation of the quality factor affected per benefit is sufficient. (4)

3 c) *Scenario*: In your phone you have installed a gym app which is part of the gym system described in problem **1b**). The system has a feature that detects when your phone comes close to a specific exercise machine. It then finds the preferences you have for that machine (e.g. height of seat, weight of resistance) and after the exercise, the app asks you if you want to adjust anything for the next time. The result is stored with your progress data, which can be monitored by yourself and your personal trainer.

Task: Draw a UML Sequence diagram of at least 3 roles and at least 10 messages that models the feature. Write down additional assumptions, if you need them for your solution. (4)

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. In Equivalence class testing you cover one, and only one, invalid Equivalence class per test case.
B. With Boundary value testing you can more easily detect typos of inequality operators in the code.
C. Equivalence class testing applies only to system and acceptance testing.
D. In Boundary value testing you don't need to identify equivalence classes of input variables.

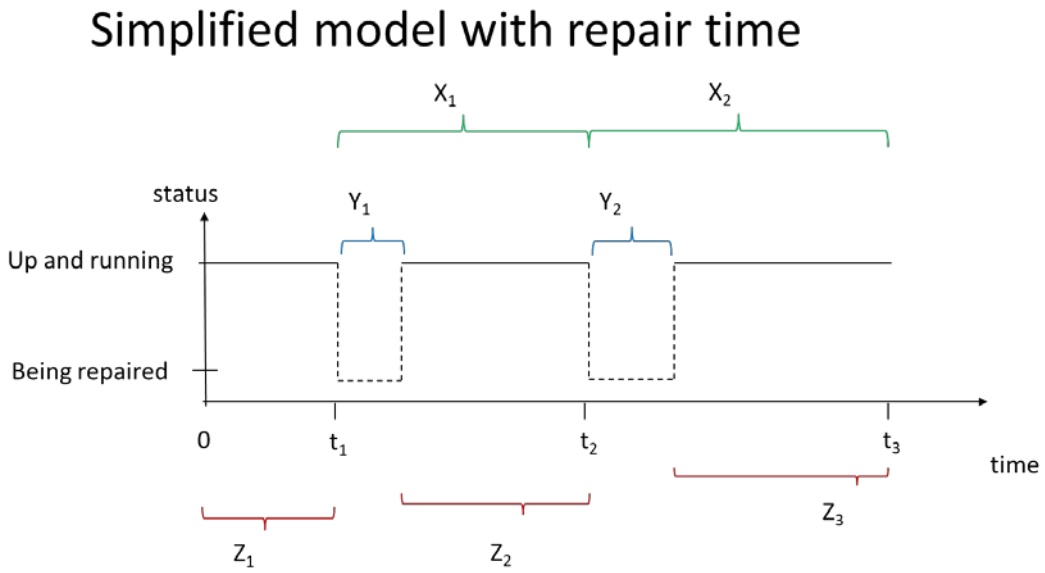
4 b) Briefly describe the workflow of a developer changing code in a decentralized modify-merge version handling tool. (4)

4 c) Describe a benefit with each of the integration testing strategies: Big-bang, Bottom-up, Top-Down, and Sandwich Testing. (4)

Area 5: Software Quality

5 a) Consider the failure model below where failures occur at time t^1, t^2,.... Which of the following statements are true? Answer with the statement letter only, no motivation is needed. (2)

A. The mean of all X's can be used (together with other metrics) to estimate availability.
B. A high value of the mean of all Z's indicates high reliability.
C. The mean of all Y's is called Mean Time To Failure.
D. The mean of all Z's is called Mean Time Between Failures.



5 b) Describe with an example the concept of a CMMI Process area. (4)

5 c) State and describe four roles involved in a software inspection process. (4)

Part 2: Advanced

6. Reflect on how to handle Boundary value testing if the variables are real numbers instead of integers. In the course, we said, "Create test cases for each boundary value by choosing one point on the boundary, one point just below the boundary, and one point just above the boundary." This applies to integers, but what happens with reals? Which general recommendations can you think of? What type of faults can we discover? What do we do if the boundary is π? Does it even make sense to use Boundary value testing for real numbers? (10)

7. Originally eXtreme Programming, XP was composed of the following rules:

The Rules of Extreme Programming

Planning

- User stories are written.
- Release planning creates the release schedule.
- Make frequent small releases.
- The project is divided into iterations.
- Iteration planning starts each iteration.

Managing

- Give the team a dedicated open work space.
- Set a sustainable pace.
- A stand up meeting starts each day.
- The Project Velocity is measured.
- Move people around.
- Fix XP when it breaks.

Designing

- Simplicity.
- Choose a system metaphor.
- Use CRC cards for design sessions.
- Create spike solutions to reduce risk.
- No functionality is added early.
- Refactor whenever and wherever possible.

Full Size Extreme Programming Project

Coding

- The customer is always available.
- Code must be written to agreed standards.
- Code the unit test first.
- All production code is pair programmed.
- Only one pair integrates code at a time.
- Integrate often.
- Set up a dedicated integration computer.
- Use collective ownership.

Testing

- All code must have unit tests.
- All code must pass all unit tests before it can be released.
- When a bug is found tests are created.
- Acceptance tests are run often and the score is published.

Suppose you are building a highly interactive software for the consumer market, for instance a mobile multiplayer game, in a team of eight skilled programmers. You are the team leader and have been asked to select five of the rules that you think would be the best for your project. State the five rules you select and for each rule, state:

a) The name of the rule.
b) A description of how it works.
c) A description of expected benefits for your project.
d) Potential drawbacks or risks of using the rule.

(20)

8. The gym system described in the scenario of task **1b)** and **3c)** also have the feature that you can open the doors and exercise during nights. The security requirements for this feature are high, just swiping a card or coming close with a mobile will not do.

Now you have a couple of loosely specified features and you are going to design a prototype of a complete gym system. You are encouraged to add features you think are needed or useful.

Task:

a) Draw an architecture diagram of the system. Describe the meaning of the different nodes and their interfaces. Motivate which view your architecture is showing. You don't need to use UML. (10)
b) Draw a UML class diagram of the concepts that are handled by the system. The most important attributes and operations shall be included. (10)