Försättsblad till skriftlig tentamen vid Linköpings universitet

Datum för tentamen	2019-10-23
Sal (4)	TER3(68) TER4(16) TERE(2) TERF(3)
Tid	14-18
Utb. kod	TDDC88
Modul	TEN1
Utb. kodnamn/benämning **Modulnamn/benämning**	Programutvecklingsmetodik Tentamen
Institution	IDA
Antal uppgifter som ingår i tentamen	9
Jour/Kursansvarig Ange vem som besöker salen	Kristian Sandahl Lena Buffoni
Telefon under skrivtiden	013-28 19 57
Besöker salen ca klockan	15
Kursadministratör/kontaktperson (namn + tfnr + mailaddress)	Veronica Kindeland Gunnarsson, 013-28 56 34 veronica.kindeland.gunnarsson@liu.se
Tillåtna hjälpmedel	2 handwritten A4 sheets. You may write on both pages A dictionary or English word-book. 2 handsrkivna A4 ark. Ni får skriva på båda sidor. Ett lexikon eller engelsk ordbook.
Övrigt	
Antal exemplar i påsen	

Försättsblad till skriftlig tentamen vid Linköpings universitet

Datum för tentamen	2019-10-23
Sal (4)	TER3(68) TER4(16) TERE(2) TERF(3)
Tid	14-18
Utb. kod	TDDC88
Modul	TEN1
Utb. kodnamn/benämning **Modulnamn/benämning**	Programutvecklingsmetodik Tentamen
Institution	IDA
Antal uppgifter som ingår i tentamen	9
Jour/Kursansvarig Ange vem som besöker salen	Kristian Sandahl Lena Buffoni
Telefon under skrivtiden	013-28 19 57
Besöker salen ca klockan	15
Kursadministratör/kontaktperson (namn + tfnr + mailaddress)	Veronica Kindeland Gunnarsson, 013-28 56 34 veronica.kindeland.gunnarsson@liu.se
Tillåtna hjälpmedel	2 handwritten A4 sheets. You may write on both pages A dictionary or English word-book. 2 handsrkivna A4 ark. Ni får skriva på båda sidor. Ett lexikon eller engelsk ordbook.
Övrigt	
Antal exemplar i påsen	

Försättsblad till skriftlig tentamen vid Linköpings universitet

Datum för tentamen	2019-10-23
Sal (4)	TER3(68) TER4(16) TERE(2) TERF(3)
Tid	14-18
Utb. kod	TDDC88
Modul	TEN1
Utb. kodnamn/benämning **Modulnamn/benämning**	Programutvecklingsmetodik Tentamen
Institution	IDA
Antal uppgifter som ingår i tentamen	9
Jour/Kursansvarig Ange vem som besöker salen	Kristian Sandahl Lena Buffoni
Telefon under skrivtiden	013-28 19 57
Besöker salen ca klockan	15
Kursadministratör/kontaktperson (namn + tfnr + mailaddress)	Veronica Kindeland Gunnarsson, 013-28 56 34 veronica.kindeland.gunnarsson@liu.se
Tillåtna hjälpmedel	2 handwritten A4 sheets. You may write on both pages A dictionary or English word-book. 2 handsrkivna A4 ark. Ni får skriva på båda sidor. Ett lexikon eller engelsk ordbook.
Övrigt	
Antal exemplar i påsen	

Försättsblad till skriftlig tentamen vid Linköpings universitet

Datum för tentamen	2019-10-23
Sal (4)	TER3(68) TER4(16) TERE(2) TERF(3)
Tid	14-18
Utb. kod	TDDC88
Modul	TEN1
Utb. kodnamn/benämning **Modulnamn/benämning**	Programutvecklingsmetodik Tentamen
Institution	IDA
Antal uppgifter som ingår i tentamen	9
Jour/Kursansvarig Ange vem som besöker salen	Kristian Sandahl Lena Buffoni
Telefon under skrivtiden	013-28 19 57
Besöker salen ca klockan	15
Kursadministratör/kontaktperson (namn + tfnr + mailaddress)	Veronica Kindeland Gunnarsson, 013-28 56 34 veronica.kindeland.gunnarsson@liu.se
Tillåtna hjälpmedel	2 handwritten A4 sheets. You may write on both pages A dictionary or English word-book. 2 handsrkivna A4 ark. Ni får skriva på båda sidor. Ett lexikon eller engelsk ordbook.
Övrigt	
Antal exemplar i påsen	

Linköpings Universitet
IDA
Kristian Sandahl

2019-10-23

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 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 2019. 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 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 a maximum of 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 statements 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. The requirement "The numerical functions must work with double-precision floating-point format." is a *functional requirement*.
B. The requirement "The operator shall be able to break the electric circuit with a stop button." is a *non-functional requirement*.
C. A *feature* is a property of a product that can be broken down to both *functional* as well as *non-functional requirements*.
D. The requirement "The operator shall be able to read sensor values with a maximum delay of 0.2 s." is a *non-functional* requirement.

1 b) *Scenario:* You are about to develop a system for a news agency. Accredited reporters use an authorization system to identify themselves and post a news item. The editor reviews all posts and can order an analyst or another reporter to double-check the contents of the post. Analysts can search the entire database of articles both for the own agency as well as agencies which databases are subscribed to. The editor approves all news that he/she considers ready to send to subscribers.

Task: Write two *use-cases* involving three different *actors*. It is sufficient if *actors* only participate in one use-case. Draw a *UML use-case diagram*. You may make additional assumptions of how the system is used if you write them down. (4)

1 c) Describe concepts: *requirements validation, unambiguous requirement, feasible requirement*, and *requirements elicitation*. Hint: 1-2 sentences per concept is OK, you may use an example. (4)

Area 2: Design and Architecture

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

A. An *implementation view* of an architecture depicts the real-time behaviour of the components.
B. With a diagram showing the *execution view* of an architecture, you can describe the interaction between components for a given use-case.
C. *An implementation* view of an architecture is often described using some kind of *UML Structure diagrams*.
D. A *deployment view* of an architecture is often described with a *UML State machine diagram*.

2 b) For each of the following two architectural styles:

- *Layered architecture* and
- *Pipe-and-Filter architecture*

describe a *quality factor* that is positively affected by each style. Also, describe a potential drawback of each architectural style. (4)

2 c) You are developing a text management system and want to analyse and visualize the following requirements:

1. A text can be a paragraph or a section.
2. A text knows its author and time of creation.
3. A section has a heading, and maybe a sub-heading.
4. A section contains at least one text.

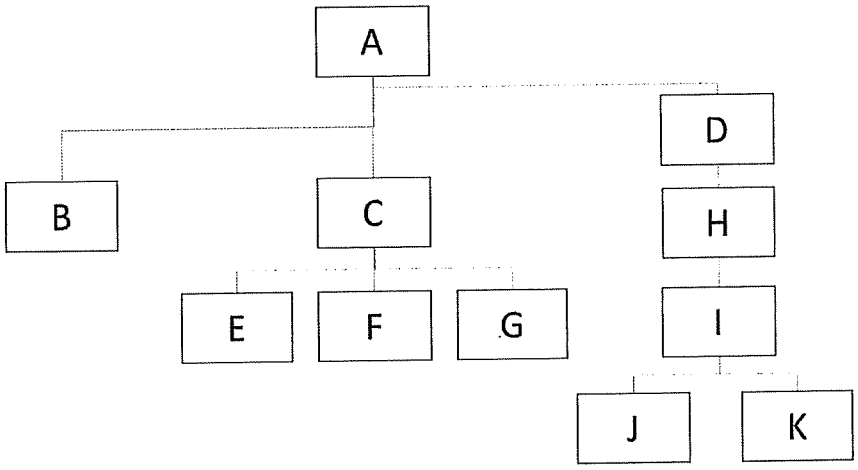
Draw a *UML Class diagram* of the elements in the requirements. Use at least one *generalization association*. (4)

Area 3: Testing and SCM

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

A. *Performance testing* can include testing a wide range of *non-functional requirements.*
B. *Pilot testing* is a kind of *Acceptance testing.*
C. *Parallel testing* is used to test *multi-threaded* applications.
D. With *control-flow coverage testing*, it is not possible to detect *missing functionality.*

3 b) Suppose you have a system with the following *functional decomposition tree.*



You are planning the *integration testing* phase of the system. You know from historical data that it takes 1 day to develop a *stub*, and 3 days to develop a *driver.* How many days will it take to prepare:

a) top-down integration testing,
b) bottom-up integration testing, and
c) big-bang integration testing?

Motivate your answers! Write also down a drawback of *big-bang integration testing.* (4)

3 c) Describe with an example the *feature branch workflow* in the distributed version-control system Git. Write down one advantage and one potential problem with the *feature branch workflow*. Don't forget to mention the workflow you are comparing to. (4)

Area 4: Planning and Processes

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

A. *Incremental methods* build components that are integrated in a piecemeal fashion with the rest of the system.
B. *Iterative development methods* focus on delivering documents, which can be reused in future projects.
C. SCRUM prescribes *pair programming* as a mandatory way of working.
D. *The classical waterfall model* requires an early commit to system architecture.

4 b) Describe the following properties of Kanban: *Kanban board, lead time,* and *work in progress*. Give one expected advantage with *Kanban*. (4)

4 c) Describe how the *Delphi method* for effort estimation works. Write down a difference between the *Delphi method* and *Planning Poker*. (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 *defect list* of an *inspection record* lists instructions to the *author* of how to fix the defects.
B. One goal of the *entry criteria* of an *inspection* process is to use the time of the *inspectors* efficiently.
C. In an *inspection*, it is good if the *inspectors* have very similar competence to give a statistical ground for *process improvement*.
D. In a *walk-through* it is the *author* who controls the discussion amongst the participants.

5 b) Describe two *metrics* that you think can give you an indication of the *maintainability* of the software. Also, motivate why it gives an indication for maintainability. (4)

5 c) Scenario: You are an enthusiastic entrepreneur who knows everything about the market and what your customers want. Your products have been on the market for two years and you hired 20 good software engineers during these years. Now as your company needs maturation you asked your personnel about the most pressing needs. They think they could do a better job at a lower cost if they all know about the current goals, sometimes there are people working with outdated goals that change very often. They also think that it is hard to know when the product is good enough for release.

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 the four phases of the *risk management process* with 3-4 sentences per phase. Give two concrete pieces of advice on how to make *risk* useful.
(10)

7. *Scenario:* A home temperature control system maintains the indoor temperature at a desired temperature (T_{set}) by controlling a heater and a cooler. The current temperature (T) is measured continuously. The system has the following requirements:

1. The user can enter a desired temperature T_{set} in the interval 18-25 °C when the system is started.
2. If both the heater and the cooler have been off for 20 minutes or more and the current temperature (T) has decreased with 2 degrees below the desired temperature or lower ($T <= T_{set} - 2$ °C), then the heater is turned on.
3. If both the heater and the cooler have been off for 20 minutes or more and the current temperature (T) has increased with 2 degrees above the desired temperature or more ($T >= T_{set} + 2$ °C), then the cooler is turned on.
4. If the cooler is on and the current temperature is 1 degree below the desired temperature or lower ($T <= T_{set} - 1$ °C), then the cooler is turned off.
5. If the heater is on and the current temperature is 1 degree above the desired temperature or more ($T >= T_{set} + 1$ °C), then the heater is turned off.
6. The cooler and the heater can never be on simultaneously.

Task: Draw a *UML State diagram* of the class temperature control system. Put all *actions* on the *transitions*, as we have done in the course. For simplicity reasons, you don't need to model the situation when you change the set temperature when the system is running. We assume that it is only done during the startup procedure. (10)

8. *Scenario:* Same as in problem 7.

Task: Create a *test table* for *Boundary value testing* of the temperature control system covering requirements 1, 4 and 5 in the scenario of problem 7. Describe the line of reasoning that you followed when you arrived at the test cases. Why did you chose the particular test cases in your test table? (10)

9. In the list below you find the *twelve principles of agile software development from the Agile manifesto*. Select five of them and show how they can be realized in SCRUM or eXtreme Programming (XP). If you take the example from SCRUM you mention which *artifacts*, *roles*, or *meetings* that are involved. If you use an example from XP you mention the *practice (rule)* that you are referring to. Your motivations shall be thorough, 5-10 sentences per *agile principle*.

For each of the selected principles also write how they can be used in a *classical waterfall model* project. If you don't think it will work, explain why. (20)

The twelve principles behind agile software development from the Agile Manifesto

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customers competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity--the art of maximizing the amount of work not done--is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly