Linköpings Universitet
IDA
Kristian Sandahl

Written exam for Software Engineering Theory

Course codes TDDC88, TDDC93, 725G64

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 2011. This gives you the mark 3 in the Swedish system. 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 in Swedish system
0-49	UK
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 statement 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 a process where you make sure that your requirements specification is following the IEEE 830 std.
B. A *feature* is a property of the system that can be decomposed into different lower level functional and non-functional requirements.
C. In UML it is not possible to have a generalization relationship between two use-cases.
D. In requirements engineering a *stakeholder* can be a person that affects, or is affected by, the system being specified.

1 b) Draw a UML *Use-Case Diagram* of a system for a social network system, such as Facebook or LinkedIn. There shall be two different use-cases and two different actors in the diagram. Use-case texts for the use-cases shall be written in complete sentences. (4)

1 c) Write down two *functional* and two *non-functional* requirements of a system storing, searching and playing streamed videos, such as YouTube or SVTPlay. (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.

2 b) Describe with an example the difference between the following pairs of risk planning approaches:

1. Risk *avoidance* – Risk *transfer*
2. Risk *mitigation* – *Contingency* plan

(4)

2 c) Below is a list from the *Agile Manifesto*. Explain the meaning of each item in the list in the context of a software development project. You may use an example:

1. Individuals and interactions over processes and tools
2. Working software over comprehensive documentation
3. Customer collaboration over contract negotiation
4. Responding to change over following a plan

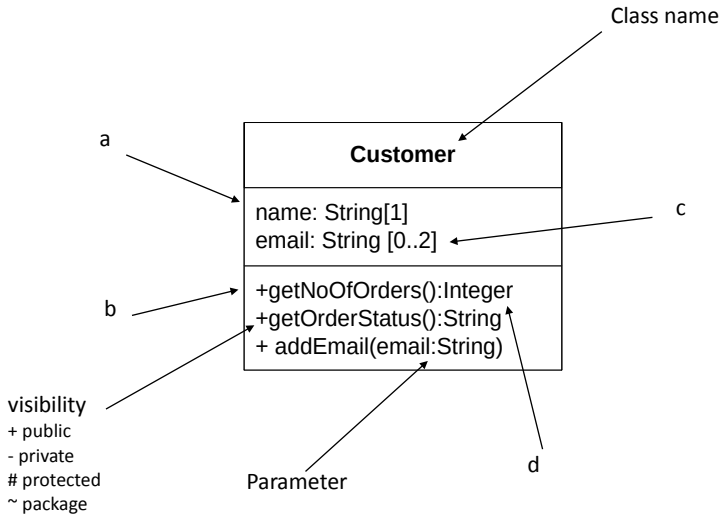
(4)

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 *layered* architecture style supports incremental development and testing.
B. A *layered* architecture style can give performance penalty.
C. A *pipe-and-filter* architecture style means that you store all your data in a central database and filter out needed information with queries.
D. A *pipe-and-filter* architecture results in a complicated dependency network that can be hard to change.

3 b) Name (in UML terminology) and give a short description of the model elements a, b, c and d in the *UML Class diagram* below.



(4)

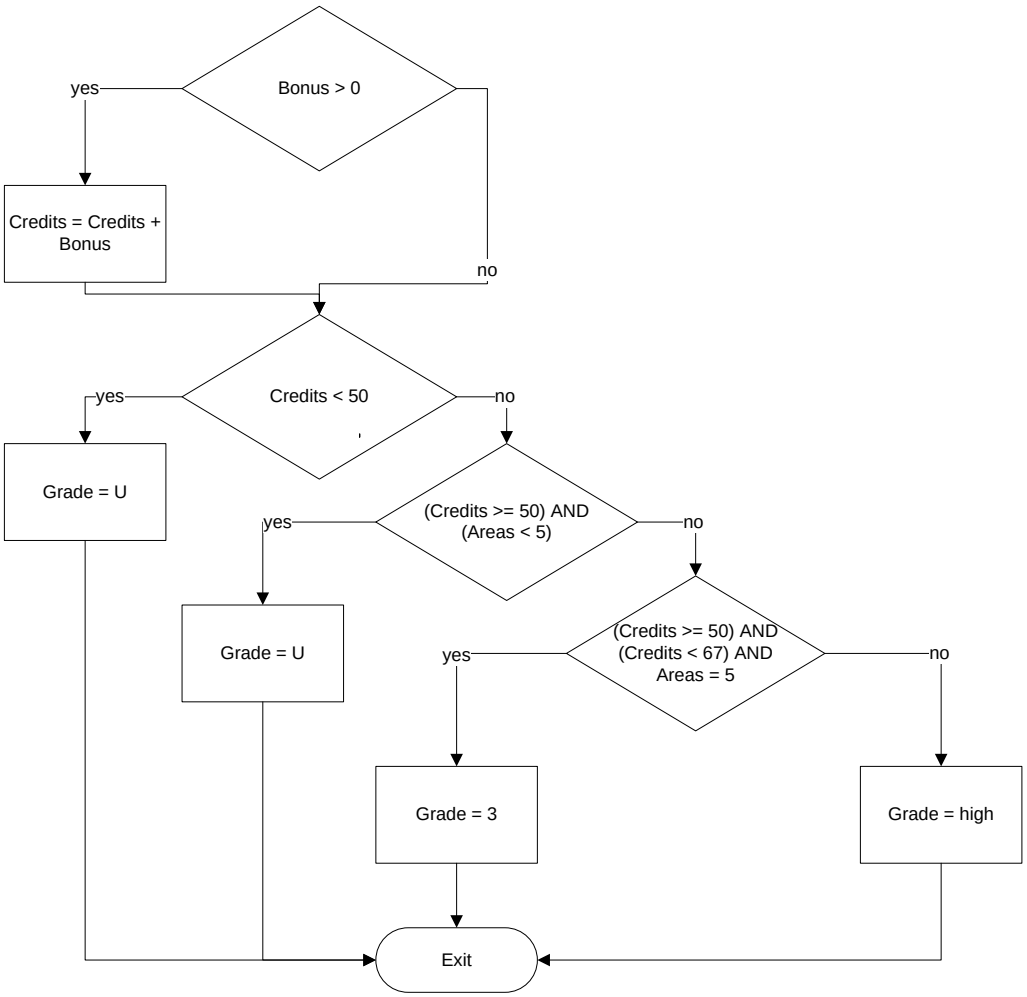
3 c) Name two different *design patterns*. For each of the patterns, describe what problems the particular pattern can solve. (4).

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. If you do integration testing according to the *bottom-up strategy* you will probably discover problems with low-level components early.
B. If you do integration testing according to the *bottom-up strategy* you will need to implement many stubs.
C. The advantage with the *big-bang integration testing strategy* is that you do not need to write a lot of stubs and drivers.
D. In *sandwich integration testing* you identify a target level and do integration testing of the modules at target level first.

4 b) Name the different parts of *a test case*, that is the information you need to specify a test case. For the *flowgraph* below write a set of test cases that ensures *statement coverage* with a minimum number of test cases.



(4)

4 c) Describe the following strategies of Configuration Management: *Lock-modify-unlock, Centralized-modify-merge, and Decentralized-modify-merge*. Name a tool for configuration management and write which of the previous strategies that the tool follows. (4)

Area 5: Software Quality

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

A. An *inspection* is a software review technique that follows a predefined, repeatable process.
B. The *author* of a document is leading the *inspection* since he/she is the main recipient of the information
C. *Inspections* require that you can engage very skilled people to do the inspection, but in return you are likely to find a majority of the total defects of your product.
D. An *audit* is a software review process where you present your product to a large audience to get feed-back.

5 b) Describe two *metrics* that can be measured on source code and that can be used to predict *reliability* of the system under development. Don't forget to motivate your answer. (4)

5 c) Describe the following concepts of the SEI CMMI: *Maturity level, process area, requirements development, and organizational process definition*. (4)

Part 2: Advanced

6. *eXtreme Programming*, XP, contains lot of practical experience and is counted as one of the most influential *agile* methods. One of the reasons for its popularity is its characteristics, which are described with a number of *practices*, also known as *facets*.

a) Write a thorough description of five of these practices. For each of the practices you should also reflect of whether the practice can be used in a traditional, water-fall inspired, method or if the practice can only be followed in an agile method. Motivate your reflections clearly (10)
b) The persons who developed XP were very clever and skilled programmers. Suppose we start with a team of newly graduated members, describe three ways in which team members can increase their competence and skill by following the XP method. (6)
c) XP and SCRUM are popular results stemming from the agile community. Make a comparison between XP and SCRUM by giving a detailed description of their differences. (4)
d) Can XP and SCRUM be used simultaneously by the same team? Motivate your answer well. (4)

7. Your company is developing a *web-based travel agency* for demanding travellers who wants complete arrangements for tailor-made tours to different countries or regions. The traveller using the system is guided through a dialogue where the user can ask for certain destinations, transportations, hotels, activities, or things to see. During the dialogue the system suggests a travelling schedule based on the input, but the system can also give suggestions, for instance interesting things to see close to the hotel, or taking trips in different order to lower the price. One of the reasons for your success is that you have a very clever algorithm that can determine the users' preferences and interests. In the back-end of the system you have a good collection of *web-services* from providers of services you combine into your customer offer. That network is also maintained by a clever selection algorithm.

a) Now it is your task to make an overall *architecture description* of the system. You don't need to use UML; *box-and-line, diagrams* will do. Describe the resulting architecture in terms of advantages and disadvantages for the quality factors: *Security, performance, usability and reliability*. (16)
b) Draw a *UML sequence diagram* with at least 5 *roles* and 10 *messages* describing an entire booking of an arrangement. The messages can be on a high level, so you don't need to take signalling and exact parameter types into account. (Hint: you probably need more than the minimum model elements to make a complete description) (10)