LINKÖPINGS UNIVERSITET 2009-01-13
IDA
Kristian Sandahl

Written exam for Software Engineering Theory

Date: 2009-01-16
Time: 08:00-12:00

Valid for pass of the written exam in courses: TDDC88 and TDDC93.

Allowed aids: Two sheets of handwritten A4 pages. You may write on both pages, with any type of size and colour. One volume of dictionary to/from English or an English wordbook.

Explicitly forbidden aids: Textbooks, machine-written pages, photocopied pages, pages of different format than A4, electronic equipment.

Graded exams will be shown 2009-01-30 at 12:30-13:30 in Conference room Donald Knuth

Questions of clarification will be answered by examiner Kristian Sandahl (013-28 19 57 or 0706-68 19 57), who will visit the exam about one hour after start.

Instruktioner till tentamensvakter

Studenterna får ha med sig 2 handskrivna A4-blad med text på båda sidorna och ett lexikon. Studenter med andra hjälpmedel utan särskilt tillstånd får inte påbörja tentamen förrän examinator kontaktats.

Instructions to students, please read carefully

- Try to solve as many problems as possible.
- Motivate all solutions.
- Please, write and draw clearly.
- Write only on one side of the paper.
- Write solutions for different areas (fundamental part) and different problems (advanced part) on separate sheets of paper.
- Label all papers with name 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 and start with the ones you directly see a solution to. This will also 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, Planning & Processes, and Quality factors. Thus the Fundamental part can give maximally 50 credits.

The Advanced part has problems worth 50 credits in total. They can be distributed over two to five problems. Each problem typically requires a longer solution of several pages.

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

Multiple choice questions will ask you to write down the label of two correct statements. Credits are given according to the following table:

Number of correct statements in the answer	Number of false statements in the answer	Number of credits
2	0	2
2	1	1
2	2 or higher	0
1	0	1
1	1 or higher	0
0	any	0

Pass condition: At least 5 credits per area in the Fundamental part **and** at least 50 credits in total. This gives you the mark 3 in the Swedish system and a C in ECTS. Credits will be added for those who passed the quiz 2008-09-19.

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	Translation to ECTS
100-84	5	A
83-67	4	B
66-50	3	C
49-0	UK	Fx

Good Luck!

Kristian

Problems

Part 1: Fundamental

Area 1: Requirements

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

1. A requirement is unambiguous if it can be traced back to its origin, for instance, a business case or an expressed customer need.
2. In requirements elicitation you use many different methods to obtain the true need of the customer.
3. The requirements "The phone shall be used in a GSM network." and "The phone shall be used in a 3G network." are inconsistent.
4. After the delivery of the system it is important to update the requirements specification in order to facilitate future maintenance and component reuse.

1 b) Draw a UML Use-case diagram and write Use-case texts for a University lecture scheduling system. There shall be at least two actors and two use-cases.(4)

1 c) Suggest four different types of readers of a requirements specification. Write a sentence or two describing what they are looking for in the requirements specification and why they need this information.(4)

Area 2: Design and Architecture

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

1. A database with applications can be regarded as a special case of a repository or blackboard system.
2. Using a pipe-and-filter architecture facilitates reuse and extension of the system.
3. A draw-back of a layered architecture is that it does not support incremental development and testing.
4. An interpreter architecture can be used to speed-up mathematical simulation software.

2 b) Suggest an application where you would like to use a two-tier architecture with a fat client. Also suggest an application where you would like to use a two-tier-architecture with a thin client. Don't forget to motivate your choices. (4)

2 c) Draw a single class UML diagram of your own choice and mark the following elements in the diagram: *Class name, attribute, visibility, return type*. (4)

Area 3: Testing

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

1) Data flow testing with program slicing is a technique for black-box unit testing.
2) The termination problem in software testing means that the system under test has gone into an infinite loop and it is hard to localise the wrongly written code afterwards.
3) Unit testing is often done by the programmers themselves.
4) Regression test means that you re-run a suite of previously run test cases to check that you get the same results for a new version of the software.

3b) Below follows a list of control-flow statement coverage criteria in alphabetical order. Arrange them in order of strength, which is the amount of test-cases needed for fulfilment. The strongest criterion shall be last in the list. Define at least one of them thoroughly with an example.

- Condition Coverage
- Decision (Branch) Coverage
- Path Coverage
- Statement/Line/Basic block/Segment Coverage

3c) Explain the terms *error, fault, failure and hazard* in the context of software engineering. (Hardware people sometimes have different definitions.) (4)

Area 4: Planning and Processes

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

1. If you input a new version of a piece of software into a Software configuration management system, you perform at commit operation.
2. The originators of RUP recommend that you spend about twice as much time in the elaboration phase compared to the construction phase.
3. The COCOMO II effort estimation method is based on academic projects only and can thus be relevant for your student projects.
4. Creating a prototype can be a way to mitigate an identified risk.

4b) Draw a small, hypothetical Gantt chart and mark examples of the following things in the graph: *task, phase, duration and predecessor*.(4)

4c) Write down two advantages and two problems with using an incremental software development method. (4)

Area 5: Quality factors

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

1. ISO 9000-3 is a guideline for how to apply ISO 9001 to software industry.
2. Total quality management means that you appoint all quality improvement initiatives to a single, experienced software engineer.
3. Function points is a language-independent method for estimating size and complexity of software to be developed.
4. Quality function deployment is a method for statistical process control.

5 b) Define the terms *level, process area, verification* and *validation* in the context of CMMI. (4)

5 c) Give four examples of data you record in the defect list of an inspection record. For each item write a sentence of why you record this data.(4)

Part 2: Advanced

6 a) It is not uncommon that software quality factors can become in conflict with each other in certain situations, for example *security* and *usability*. This can be exemplified with several arguments such as:

- Security routines take time that increases the response time, which reduces efficiency, which is a usability sub-factor.
- If we require complex passwords we increase the probability that the user will forget them and spend time in getting new ones from the system administrator. This will damage the efficiency and attitude towards the system. This will contribute negatively to usability.

Your task is now to construct at least 5 conflicting software quality attribute pairs and for each pair give an example of how you can argue that they are in conflict with each other. (10)

6 b) Software quality factors can be measured more or less directly with several different *metrics*. For instance, reliability can be measured with:

- Mean time to failures (MTTF) according to the formula MTTF/(1+MTTF). The argument is that the longer it takes between failures, the better the reliability. This is a very direct metric.
- Lines of code. The argument is that it is often hard to understand a large portion of code; the fault density is often higher for large modules. High fault density leads to bad reliability. This is a fairly indirect metric.

Your task is now to construct at least 5 triples of software quality factor, possible metric and an argument. (10)

7) You are designing software for a car rental agency. The system covers all aspects of the management of the agency including:

- Technical status of the cars
- Service intervals
- Selling used cars
- Buying new cars
- Reserving cars for customers
- Scheduling cars to reduce passive days
- Dimensioning the fleet for a good service level
- Rental contract management
- Payment management
- Car position management
- Fraud detection
- Etc.

Start sketching the system by doing the following:

- **a)** Draw a use-case diagram of two different actors and four use-cases. Don't forget the use-case texts. (5)
- b) Draw a class diagram of some part of the system, or a high-level model of the entire system. Use at least 5 classes with association, generalisation, and composition. All classes shall have at least one attribute and one operation. Make sure to get a sensible multiplicity on the associations. (10)
- c) Draw a UML state diagram of a Car. Use at least three states and 2 transitions. (5)
- d) Draw a UML activity diagram for a customer rental process. Use at least 5 activities, 1 decision and a fork or synchronisation bar. (5)
- e) Write 5 functional test cases for a part of the system with input and expected output. (5)