Written exam for Software Engineering Theory

Date: 2008-10-22
Time: 14.00-18.00

Valid for pass of the written exam in courses: TDDC01, TDDC06, TDDC88, 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 in about two weeks. Date, time and place will be announced on the homepage.

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

Instruktioner till tentamensvakter
Studenterna får ha 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 and your Swedish personal 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:

<table>
<thead>
<tr>
<th>Number of correct statements in the answer</th>
<th>Number of false statements in the answer</th>
<th>Number of credits</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2 or higher</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1 or higher</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>any</td>
<td>0</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>Total credits</th>
<th>Mark in Swedish system</th>
<th>Translation to ECTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>100-84</td>
<td>5</td>
<td>A</td>
</tr>
<tr>
<td>83-67</td>
<td>4</td>
<td>B</td>
</tr>
<tr>
<td>66-50</td>
<td>3</td>
<td>C</td>
</tr>
<tr>
<td>49-0</td>
<td>UK</td>
<td>Fx</td>
</tr>
</tbody>
</table>

**Good Luck!**

**Kristian**
Problems

Part 1: Fundamental

Area 1: Requirements

1a) Which of the following statements are true? Answer with the statement number only. No motivation is needed. (2)
   1. It is practically impossible to transform the information in an Entity-Relationship-diagram into UML Class diagrams.
   2. Requirements classification can be used to prioritize different requirements.
   3. An actor in a UML Use-case diagram can be either a human user or another system.
   4. The requirement “The system shall be ready to use 15 minutes after the start-up is initiated.” is a functional requirement.

1b) Write down two functional requirements and two non-functional requirements of a mobile telephone. (4)

1c) Describe the goal of Requirements Elicitation and give two examples of techniques that can be used in Requirements Elicitation. Give an example of a stakeholder, other than the Requirements analyst, that is heavily involved in Requirements Elicitation. (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. In a Two-Tier Client-Server Architecture with thin clients, keeping the software up to date on the clients is a major issue.
   2. In a Three-Tier Client-Server architecture you can distribute the system components on three different hardware nodes.
   3. When we say we want low coupling, we are thinking of making future changes cheaper.
   4. In design it is important to always throw prototypes away, since we don’t want to risk that preliminary thinking comes into the final design.

2b) Explain the terms: Divide and conquer, data abstraction, encapsulation and polymorphism. (4)

2c) In a consultant time reporting software, the consultants enter their worked hours per week specified in project numbers and if overtime applies. The customers and project leaders can verify their reports and the reports are sent both to a payroll system and an invoice system. Draw a UML class diagram of at least 4 classes. Each class shall have at least one attribute and one operation. No class is left without association to at least one of the other classes. (4)
Area 3: Testing

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

1. Stress or overload testing means that you observe the end-user's behaviour using the system in a stressful working situation.
2. Beta testing is the second suite of test cases you run in System Testing
3. JUnit is a tool for unit testing that is good to combine with a bottom-up integration testing strategy.
4. Using Equivalence class black-box testing probably means that more code statements are covered compared to randomly selecting test cases.

3b) In the list below you find different results from a data flow test. Pick out four of them that are correct and/or acceptable. Give a short motivation for each of the four results you chose. (4)

- dd: defined and defined again
- du: defined and used
- dk: defined and then killed
- ud: used and defined
- uu: used and used again
- uk: used and killed
- kd: killed and defined
- ku: killed and used
- kk: killed and killed

3c) Explain the terms oracle, stub, driver and parallel testing. (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. Managing a department at a university can not be regarded as a project, since there is generally no time limit of when the department's mission shall be completed.
2. The Delphi method can be used both for estimating costs and setting the priorities of the work packaged in a project.
3. Slack time of an activity means that the persons involved in the activity are not booked and that they can take some days off for education and training.
4. In test-driven development, the test leader acts as project leader

4b) Write down a standard agenda for a weekly project meeting with all participants. At least four of the items on the agenda shall have a short motivation of why they are listed. Opening and closing the meeting can be taken for granted and can not be selected amongst the four motivated items (4)

4c) The single round Waterfall model has often been criticised. Make a more balanced view by writing down two benefits and two drawbacks of the Waterfall model. Don't forget to add a short motivation. (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. Risk management is necessary for an organisation at CMMI level 3.
2. TickIT is a certification organisation determining the CMMI level for organisations.
3. If your organisation doesn't fulfil 2 process areas on CMMI 2 and 4 process areas on CMMI 3, it is advised to start with the ones belonging to CMMI 2, even if the goal is to reach CMMI 3.
4. ISO 9000-3 is built on the principle of democratic system development where each participant has a vote on the project meetings.

5b) Suggest four software metrics for maintainability and/or usability. It shall be possible to give a single number as a measurement. Specify for each metric how it can be calculated by explaining how data can be obtained and how you use data in calculating the metric. (4)

5c) Apart from the defect list you can obtain interesting data from the inspections. Give three examples of data you can obtain, and give one example of how you can use that to improve your business. (4)
6) In the table below we have listed some life-cycle and process models and a few properties. Your job is to fill in the blank combinations by writing a motivated statement of whether you think that the life-cycle or process model has the property or not. You receive 1 credit per sensible statement, max 20 are counted (20)

<table>
<thead>
<tr>
<th>Style/Property</th>
<th>The original waterfall model according to Royce (1970).</th>
<th>The incremental model.</th>
<th>The Spiral model.</th>
<th>An agile model, such as XP or Scrum.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Suitable for large projects.</td>
<td>Yes. In large projects it is important to have well-defined phases and documents, since it is hard to have everyone at the meetings.</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Suitable for fixed-price projects.</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Suitable for safety-critical systems.</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Suitable for completely new systems where things will be changed.</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Suitable for parallel development of independent teams.</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Suitable for a student project.</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
7) Appended are David's slides on configuration management and configuration management tools. You are the designer of a new configuration management tool. 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. Make sure to get a sensible multiplicity on the associations. (5)
- c) Draw a UML state diagram of a Configuration item. Use at least three states and 2 transitions. (5)
- d) Draw a UML activity diagram for a change control process. Use at least 5 activities, 1 decision and a fork or synchronisation bar. (10)
- e) Write 5 functional test cases for a part of the system with input and expected output. (5)
What is configuration management?

- **Configuration Item Identification**
  - Source code modules
  - Test scripts
  - Design documents
  - Build systems

- **Configuration control**
  - Larger changes - change requests

- **Auditing**
  - Ensure that the items are complete and consistent.
  - Make sure that the configuration is tested and meets requirements.

- **Status accounting**
  - Document and report changes to those involved.
SCM tools

Change Management

Workflow systems
- Define processes for change requests

Change report system
- Bug-tracking
- New features
- Change request
(e.g free alternatives: Bugzilla, Trac)

What is a bug / what is a feature?

Tool examples:
- Clear Case,
- Visual Source safe
- Perforce,
- CVS
- Subversion

Locked checkout / no locks

Commit - add comments
History - Find where the fault was added
Blame - Who added a certain code line
Diff - What has changed between two versions?