Written exam for Software Engineering Theory

Date: 2008-01-15
Time: 08.00-12.00

Valid for pass of the written exam in courses: TDDB61, TDDB62, TDDC01, TDDC06, TDDC88, TDDC93.

Allowed aids: Two sheets of handwritten A4 pages. You may write on both pages, with any type of size and color. 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 handed out 2008-01-31 in Donald Knuth (IDA) between 12.30 and 14.00.

Questions of clarification will be answered by Kristian Sandahl (070-6681957), who will visit the exam about one hour after start. Examiner David Broman can be reached on phone 0707-909075 during the exam.

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 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 Kristian questions about all parts of the exam, since he 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.

**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 an E in ECTS. Note that no credits are added, even if you passed the quiz exam in 2007.

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>
</tr>
</thead>
<tbody>
<tr>
<td>100-84</td>
<td>5</td>
</tr>
<tr>
<td>83-67</td>
<td>4</td>
</tr>
<tr>
<td>66-50</td>
<td>3</td>
</tr>
<tr>
<td>49-0</td>
<td>UK</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Total credits</th>
<th>Mark in ECTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>100-96</td>
<td>A</td>
</tr>
<tr>
<td>95-83</td>
<td>B</td>
</tr>
<tr>
<td>82-68</td>
<td>C</td>
</tr>
<tr>
<td>67-55</td>
<td>D</td>
</tr>
<tr>
<td>54-50</td>
<td>E</td>
</tr>
<tr>
<td>49-0</td>
<td>F</td>
</tr>
</tbody>
</table>

**Good Luck!**

**David & Kristian**
Problems

Part 1: Fundamental

Area 1: Requirements
1a) It is often said that it is good to supplement requirements with a link to a business case, which describes the economic rationale for the requirements. Write down two potential benefits from this. (2)

1b) Write down two functional and two non-functional requirements for a GPS (Global Positioning System) receiver for a car. (4)

1c) Write two use-cases and draw the UML use-case diagram for a railroad ticket booking system. (4)

Area 2: Design and Architecture
2a) Describe the difference between swimlanes and lifelines in UML. Give examples of UML diagrams illustrating both concepts. (3)

2b) Assume that you are developing a graphical word processor application in C++. Each time a user presses <ctrl>-S (control S), the document should be saved. However, depending on the selected file format, different document output formats should be generated and saved (e.g. HTML, word doc, or plain text). It should be possible to change file formats dynamically.
   - Suggest and motivate a suitable design pattern for this task. (2)
   - Draw a UML class diagram describing the pattern in the context of the word processor application. Describe the diagram and explain why polymorphism enables the dynamic behavior of the solution. (3)

2c) Name 4 quality factors which can affect architecture decisions. Choose one of the quality factors and describe how it can affect the architecture decision. (2)
**Area 3: Testing**

3a) A product manager states:

"When the developers have executed the *acceptance test*, the customer can perform a fast *unit test* on all installed modules in his own environment. Finally, our test department can use *regression testing* while performing the *system test* in our environment. The regression is especially useful to find bugs which have been there in several released versions of the software."

- Motivate what is correct and what is wrong in the above statement. (2)
- Give clear and correct definitions for each of the tests printed in italic above (3)

3b) Which are the main differences between formal verification (using formal methods) and testing of software? Give at least one advantage or one drawback for each of the approaches. (3)

3c) Describe the meaning of *code coverage* in a testing context. Motivate if this is an example of white-box testing, black-box testing, or neither black- or white-box testing. (2)

**Area 4: Planning and Processes**

4a) Effort estimation in software projects is difficult; especially in the beginning of a project. Give at least 3 good reasons for why this is the case. (3)

4b) Explain and give examples for the concepts of *incremental development, time-boxing*, and *iterative development*. Discuss how the concepts relate to each other. (3)

4c) Write down a *table of contents* for a typical project plan. Give a short description of what each chapter should contain and the purpose of it in a software project. (4)
Area 5: Quality factors

5a) Software inspections can be carried out in two different variants:
   - individual Preparation – group Detection (PD)
   - individual Detection - group Collection (DC).

Explain the two variants. Which one do you think makes the most efficient use of your personnel? Give an argument for your selection. (3)

5b) Select four quality factors. For each of the quality factor provide a name, definition and suggest a measurement for the quality factor. (4)

5c) Below you find a record from failure occurrences when testing a system. We can assume that once a failure is found, it is corrected.

<table>
<thead>
<tr>
<th>Failure number</th>
<th>Failure time [hours execution time]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>300</td>
</tr>
<tr>
<td>2</td>
<td>570</td>
</tr>
<tr>
<td>3</td>
<td>960</td>
</tr>
<tr>
<td>4</td>
<td>1290</td>
</tr>
<tr>
<td>5</td>
<td>1740</td>
</tr>
<tr>
<td>6</td>
<td>2100</td>
</tr>
<tr>
<td>7</td>
<td>2640</td>
</tr>
<tr>
<td>8</td>
<td>3090</td>
</tr>
<tr>
<td>9</td>
<td>3750</td>
</tr>
<tr>
<td>10</td>
<td>4500</td>
</tr>
<tr>
<td>11</td>
<td>5070</td>
</tr>
<tr>
<td>12</td>
<td>5970</td>
</tr>
<tr>
<td>13</td>
<td>6930</td>
</tr>
<tr>
<td>14</td>
<td>7680</td>
</tr>
<tr>
<td>15</td>
<td>8880</td>
</tr>
</tbody>
</table>

Calculate an approximation of the failure intensity per 1000-hour interval from 0 and 9000 hours. Draw a table. According to your experience, write down a rough estimate of the failure intensity of a light bulb. Is the system better or worse than the light bulb according to your estimate? Write down a short answer. (3)
Part 2: Advanced

6) Imagine that you have just finished your university degree and are now employed in the small software development company BookIT AB. Some facts about the company:

- It has 20 employees, including 12 developers and 1 software development manager.
- It develops an off-the-shelf software for booking (e.g. for room booking, personnel etc.).
- The skills of the developers are fairly good, but they do not use any well-defined software development process or tools. The process they are using right now looks like the waterfall model.
- They give out a new release 2 times per year, but the management team is not satisfied and wants releases more often.
- The sales and marketing employees continuously come with new feature requirements.
- Customers send in e-mails about problems they have. Developers fix the problems in new versions (if they happen to read the email). However, sometimes already fixed problems reoccur in new versions.

Since you are new to the company, you want to give a good impression and therefore you decide to write an e-mail to the software development manager, giving suggestions for improvements (maybe not the best thing to do, but you do it anyway :-)).

Write down the e-mail containing the following information:

a) Motivate why it is good to have a well defined software process and argue that the waterfall model is not suitable for this company. Describe and motivate an alternative approach to the waterfall model. (5)

b) Choose a software development process that you want to recommend for the company. Explain the process in a detailed and concrete way, so that the manager understands how it works. Make sure to discuss both pros and cons of the suggested development process. (10)

c) Describe what the following concepts / tools mean and why they would be good to introduce in the company. Discuss the cost in effort it would require to introduce them and suggest and motivate an order in which they should be introduced.

- Continuous integration
- Software Configuration Management (SCM) Tool
- Bug tracker software
- Test-first programming (test-driven programming)
- Daily Build

(15)

Remember that you need to be humble and polite in your email, since you have only worked there for a couple of weeks.
Imagine a system for management of large engines, for instance, on a ship, with the following features:
- Real-time control and monitoring of more than 100 variables.
- Configuration support for different operator views.
- Routines for automatic start-up and turn-off.
- Alarm handling and emergency prioritising.
- Routines for moving to safest possible state.
- Statistics of fuel, oil, steam and water consumption and production.
- Automatic scheduling and warning system for maintenance.
- Management of maintenance documentation and reports.
- Configuration support for optimizing run conditions.
- Voice, picture and data communication amongst personnel through hand-held devices.

You have been given the responsibility for implementing such a system from scratch! This is a rare opportunity, so you want to do this carefully by the book.

a) Describe at least 5 major components (artifacts) of such a system; draw a UML deployment diagram of at least 3 nodes where the components are placed. Write down a short motivation for your deployment. (5)

b) For one of the possible components, write at least 5 use-cases and draw a UML use-case diagram. At least two use-cases need to have more than one actor associated with it. This does not necessarily have to be a component of the problem above. (5)

c) Write an overall test plan for such a system consisting of at least 5 components. The plan shall include:
- A description of suitable methods, tools and techniques for testing of each component.
- A schedule for testing of the individual components, and for the integration testing. Try to do as much as possible in parallel. The entire software development phase can be estimated to 1.5 years.
- An estimation of the amount of personnel and test cases needed for testing the components. Motivate your estimation. It is important that your relative estimations are sound, so that a more complex component is tested more carefully than a less complex component. The absolute values are of course hard to estimate, but you shall provide some kind of reasoning. For instance, a modern high-speed ferry for 2000 passengers costs around 1 000 000 000 SEK. How much of this can be attributed to software testing? (10)