Linköpings universitet
IDA Department of Computer and Information Sciences
Doc. Christoph Kessler

EXAM

FDA149 / TDDC54 **Software Engineering**

2006-05-29, 09:00–13:00

Name: ______________________ **Personnummer:** ______________________

Please mark solved problems with X:

Question	1	2	3	4	5	6	7	8	9	10	11
answered (X)											
score (points)											

Total score: ______________________ **Grade (U/3/4/5):** ______________________

Sign. Examinator: ____

Examinator: Christoph Kessler

Jour / tentavakt (Linköping): Christoph Kessler (070-3666687, 013-282406)

External supervisors for exams written outside Linköping,

please confirm with your signature below that you have checked the identity of the candidate and supervised this exam 10/2/2006 from 14:00–18:00 before you send it by mail to Christoph Kessler, IDA, Linköpings universitet, 58183 Linköping.

Supervisor (name, signature): ______________________ ______________________

Hjälpmedel / Admitted material:

None.

General instructions

- This exam has 11 assignments. Read all assignments carefully and completely before you begin.
- It is recommended that you use a new sheet for each assignment. Number all your sheets, and mark each sheet on top with your name, personnummer, and a page number.
- You may answer in either English or Swedish.
- Write clearly. Unreadable text will be ignored.
- Be precise in your statements. Unprecise formulations may lead to a reduction of points.
- Motivate clearly all statements and reasoning.
- Explain calculations and solution procedures.
- The assignments are *not* ordered according to difficulty.
- The exam is designed for 40 points. You may thus plan about 5 minutes per point.
- Grading: U, 3, 4, 5. The preliminary threshold for grade 3 is 20 points.

Assignments

1. (6 p.) **Software processes**

   (a) Write down two drawbacks of deploying a prototyping process model. (2p)

   (b) Write down two rules of *eXtreme Programming*. (2p)

   (c) Suppose your customers are sending you ideas to new requirements continuously. Select a life-cycle model that is appropriate for this situation. Write down the life-cycle model together with a motivation. (2p)

2. (4 p.) **Software project organization**

   (a) There are many models and methods for software development effort estimation. Write down the name of one of them. Write down a motivation for your choice and list the data you would need to use the model. (2p)

   (b) Write down two arguments for using inspections as a means to validating software. (2p)

3. (2 p.) **Testing**

   Suppose a program contains N decision points, each of which has two branches. How many test cases are needed to perform path testing on such a program? If there are M choices at each decision point, how many test cases are needed for path testing? Can the program's structure reduce this number? Give an example to support your answer. (2p)

4. (5 p.) **OO technology**

   (a) Class A, which features a single, externally visible method `m` with the signature

   ```
   Type0 m ( Type1 param1, Type2 param2 ),
   ```

   has been used in a large legacy software system for years but became too slow for the increased throughput and should be replaced. There are several faster candidates on the component market that promise to be able to replace A. Apart from performance aspects, how can we make sure that a class B "fits" as a replacement of A? Formally speaking, what is the condition for a class B to be able to substitute class A in all its uses (syntactic substitutability)? (2p)

   (b) Explain the *Syntactic Fragile Base Class Problem*. What are its implications for the compatibility of binary components compiled from OO programs? (3p)

5. (1 p.) **Metaprogramming**

   (a) Define the term *introspection*, and give an example of where and how it is used in some component system. (1p)

6. (4 p.) **Design patterns and UML**

   (a) What is a *design pattern*? (1p)

   (b) Describe the *Observer* design pattern, both in structure and behavior. Give both a short textual description and suitable UML diagrams. (If you do not recall the exact UML notation, invent your own one but explain carefully what each symbol means.) (3p)

   If you do not remember the Observer design pattern, you can *instead* describe *any* design pattern of your choice (give its name, textual and UML description as above). (2.5p)

7. (3 p.) **Model-driven architecture (MDA)**

   (a) Give a short description of Model-driven architecture (MDA) (main ideas). (2p)

   (b) Explain how MDA supports reuse. (1p)

8. (3 p.) **Enterprise JavaBeans (EJB)**

   (a) Name two advantages of using Enterprise JavaBeans (EJB) when developing Java business applications. (2p)

   (b) One could argue about whether Enterprise JavaBeans are object oriented or not. What are the arguments against? (1p)

9. (6 p.) **CORBA**

   (a) Without middleware support, a program written in programming language A usually cannot directly call a method written in a different programming language B even if they would execute on the same computer. Give one of the technical reasons. (0.5p)

   (b) How does CORBA achieve programming language transparency for static calls? (Describe the principle and those main parts involved in CORBA static calls that are primarily relevant for enabling language transparency.) (3p)

   (c) What is an interoperable object reference (IOR) in CORBA? What is its purpose? What data does it contain? Where are IORs created? Name at least one possibility of how a client can get hold of an IOR. (2.5p)

10. (4 p.) **Software Architecture Systems**

    (a) Software architecture systems are said to be a first step towards separation of concerns. Which concerns are separated, and what are the advantages of this? (2p)

    (b) Define the term *layered architectural style* (also known as *onion architectural style*) and give an example of a software system with this style. (2p)

11. (2 p.) **Aspect-Oriented Programming**

    (a) Aspect oriented programming simplifies software development by providing a mechanism for implementing concerns that crosscuts modules. Describe a major disadvantage with aspect oriented programming. (1p)

    (b) What is an *advice* in aspect-oriented programming? (1p)