MTAT.03.094 – Software Engineering

Written Exam – 09 January 2015

Start: 14:15 – End: 16:15

Important Notes:

- The exam is open-book and open-laptop. Web browsing is allowed, but you are not allowed to use e-mail clients nor Instant Messaging clients, nor to share any information “live” with anybody inside or outside the exam room.
- At the end of the exam you must submit both the question sheets and your answer sheets. To avoid that any of your solutions get lost, make sure to write your name (and student ID) on each sheet of paper that you submit. Also, please number the pages on your answer sheets.
- Write clearly. Answers that are illegible cannot be counted as correct answers. Only answers written in English will be marked.
- Total marks: 30 (equivalent to 30% total grade). You must get at least 10 marks in this exam to not fail the course. There are 5 bonus marks available.

==========

PART 1: Multiple-Choice Questionnaire (10 marks + 2 bonus marks)

PART 2: Open Questions (10 marks + 3 bonus marks)

PART 3: Constructive Tasks (10 marks)

Total: 30 marks (=100%) plus 5 bonus marks

PART 1: Multiple-Choice Questionnaire (10 marks + 2 bonus marks)

Important: For Part 1, please check boxes on the separate questionnaire answer sheet. Read carefully before you answer and observe instructions carefully!

The following questions (up to question Q-10) have exactly one correct answer, thus, you must check exactly one answer box on the separate answer sheet. If you think that more than one answer is correct, choose the one answer that seems to be most correct/suitable/relevant.

Question Q-11 at the end of Part 1 is a bonus questions.

Q-01 (1 mark): Which of the following statements describes work that is not part of the (core) work of a software engineer?
Answer choice:
A: To understand the problem domain of a software system to-be-developed
B: To define/evolve the architecture of a software system
C: To design and develop program code
D: To invent a new programming language

Q-02 (1 mark): What are Use Case Points being used for?
Answer choice:
A: To estimate project effort
B: To estimate project quality
C: To estimate project duration
D: To specify requirements

Q-03 (1 mark): Which of the following statements about UML class diagrams is not correct.
Answer choice:
A: A class has the following elements: name, list of attributes, list of operations
B: Inheritance is a specific type of composition relationship
C: Composition describes a whole/part relationship
D: A superclass and its subclasses describe a generalization relationship

Q-04 (1 mark): Which of the following statements about use case descriptions is not correct
Answer choice:
A: A participating actor is an actor who helps achieve the goals of the initiating actor
B: An alternate flow describes exeptions from or extensions to the normal interaction scenario
C: A use case diagram is a graphical representation of a use case description
D: Preconditions describe the state of the system before the start of the interaction scenario

Q-05 (1 mark): Why is requirements elicitation difficult?
Answer choice:
A: Because there doesn't exist a suitable UML notation for requirements elicitation
B: Because it is difficult to identify the relevant stakeholders, and, once identified, the stakeholders have difficulties describing what they want/need
C: Because requirements can change over time
D: Because stakeholders don't understand use cases

Q-06 (1 mark): Which of the following statements about code refactoring is correct?
Answer choice:
A: Before refactoring a code module, a test suite for this code module must be in place
B: Refactoring is done in order to add new functionality
C: Refactoring is a test activity
D: Refactoring does not influence (or change) the program design

Q-07 (1 mark): Which of the following statements about test techniques is correct?
Answer choice:
A: Mutation testing does require a test oracle derived from the program specification
B: Equivalence class partitioning is a white-box testing technique
C: White-box testing techniques exploit knowledge about the code that is tested
D: Black-box testing techniques exploit knowledge about the code that is tested

Q-08 (1 mark): You execute a program and you see the following message on the screen: 'Program aborted due to division by 0'. Which of the following terms describes your experience best?
Answer choice:
A: I made an error (while using the program)
B: I debugged an error
C: I located a fault
D: I triggered a failure

Q-09 (1 mark): Which of the following types of meetings is not defined by SCRUM?
Answer choice:
A: Product Owner Meeting
B: Sprint Retrospective
C: Daily Scrum
D: Sprint Review

Q-10 (1 mark): Which of the following is not a suitable unit for effort measurement?
Answer choice:
A: Person-Day
B: Use Case Point
C: Man-Month
D: Developer-Hour

The following question (question Q-11) is a bonus question and can have more than one correct answer. You must check all correct answer choices to get full marks. You get partial marks, if you check some of the correct answer choices. You will get a penalty, if you check an incorrect answer choice. You don't get a penalty, if you miss a correct answer choice. Overall, the lowest possible mark you can get is 0 (i.e., even if everything you check is wrong, you won't get a negative mark).

Q-11 (2 marks - bonus): Which of the following statements about MVC (Model-View-Controller) are correct?
Answer choice:
A: MVC is a domain model pattern
B: MVC is an architectural pattern
C: MVC is not a three-tiered pattern
D: MVC is a reference architecture pattern

PART 2: Open Questions (10 marks + 3 bonus marks)

Note: Please give your answers on separate answer sheet(s) and state clearly to which question number each answer refers. Answers that have no question number stated will not be marked. Don't forget to write your name on each separate answer sheet.

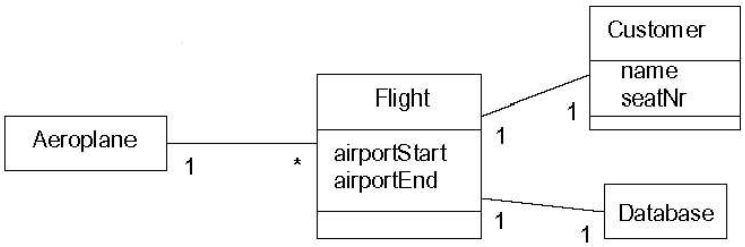
Q-12 (3 marks): Briefly explain the most important differences between 'Software Engineering' (SE) and 'Software Development in a Craftsmanship-style' (SDC). Do this in two steps:

- Step 1: List/Describe two essential elements (or principles) that are dominant in SE but not very common in SDC. Give an example for each element (or principle) listed/described
- Step 2: Describe two typical characteristics of projects where following the SDC approach might be justified (as compared to projects where following an SE approach would be recommended).

Q-13 (3 marks): Assume you observe a programmer first writing a piece of code and then testing the same piece of code. First make up one example of a 'fault' and one example of a 'failure' (in the context of this example). Then explain – in your hypothetical example – how the 'fault' was introduced and how it was localized. Finally, explain what caused the observed 'failure' and how the 'failure' was triggered.

Q-14 (2 marks): The domain model shown below describes a part of a booking system of an aviation company. The model contains several faults. Find at least 2 faults and explain for each fault what the problem is.

- Hint: Check for plausibility of restrictions (i.e., multiplicities in associations), choice of attributes, and choice of domain classes.



Q-15 (2 marks): Assume a software development process that makes the two statements shown below. For each statement, identify the type of waste the statement might trigger. In your answers, it is sufficient to state one of the seven types of waste defined in the context of 'lean software development'. For each statement, chose the <u>one</u> type of waste that is most obviously related to it.

- Statement 1: A product owner might have to be involved in several concurrent product development activities at a time.
- Statement 2: The software design document shall emerge from contributions of at least five different stakeholders: architect, product owner, project manager, customer, and developer. The contributions from the various stakeholders to the design document may be added in turns and in several iterations.

Q-16 (3 marks – bonus): Assume, you want to measure <u>productivity of your development team</u>. Typically, productivity is measured in terms of output per input.
<u>To do:</u> Define your productivity measure by specifying the following:

(1) Output entity that you are going to measure
(2) Attribute of the output entity that you are going to measure
(3) Unit of the output measure
(4) Input entity you are going to measure
(5) Attribute of the input entity that you are going to measure
(6) Unit of the input measure

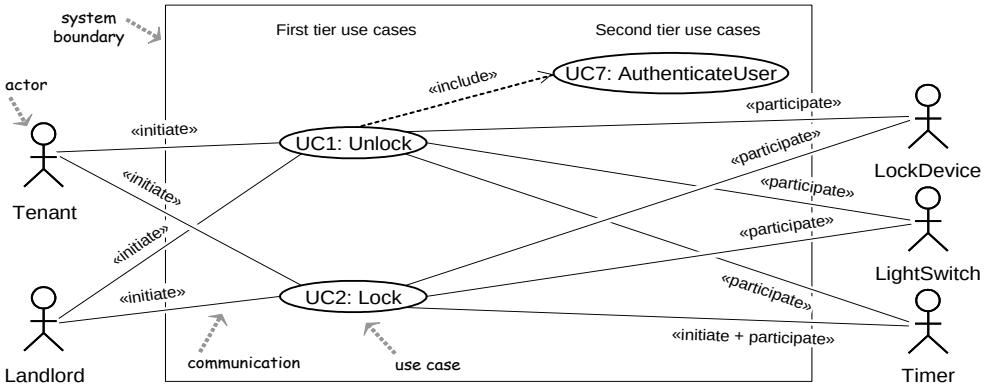
PART 3: Constructive Tasks (10 marks)

Note: Please provide solution on separate answer sheet(s) and state clearly to which task number each solution relates. Solutions that have no task number stated will not be marked. Don't forget to write your name on each separate answer sheet.

T-01 (3 marks): Make the following assumptions for the elements in the use case diagram shown in the figure below:

- Actors 'Tenant' and 'Landlord' are of type 'complex', all other actors are of type 'average'
- UC1 and UC7 have 2 transactions each
- UC2 has 4 transactions
- Technical and Environmental Complexity Factors both equal '1'
- Assume that all eight Environmental Factors equal '3'

Calculate the Unadjusted Use Case Points (UUCP), use Case Points (UCP) and Project Effort for the use case diagram in the figure below. Show all your calculations. If you only write down end results without showing the formulas used and calculations made, you won't get marks.



T-02 (5 marks): The code snippet below shows a method that calculates fines for speeding when driving a car. Fines are calculated based on age of the driver, overspeed, and number of penalty points (licencemark) accumulated in the driver's police record.

```
1  public static int speedingfine (int age, int overspeed; int licencemark) {
2    int fine = 0;
3    if ((age >= 25) && (overspeed < 30) && (licencemark < 3)) {
4       fine = fine + 100 * overspeed;
5       return fine; }
6    if ((age < 25) || (licencemark >= 3))
7       fine = fine + (200 * overspeed);
8    if (overspeed >= 30)
9       fine = fine + 5000;
10   return fine; }
```

To do:

(1) Calculate the Cyclomatic Complexity (McCabe). Show the details of your calculations.

(2) Design a set of test cases that will cover all linearly independent paths in the CFG. Remember that complete test cases include both input values and expected output values.

Hint: For (2) use the following table to present your test cases (see next page):

Test case number	Age (in)	Overspeed (in)	Licencemark (in)	Fine (out)
1				
2				
…				

T-03 (2 marks): Point out the code smells in the code shown below. Then briefly describe how you would refactor the code. Note: You don't need to write down the refactored code. It is sufficient, if you briefly say how you would refactor the code to remove the identified code smells from it.

```
class BookRental {
   String bookTitle;
   String author;
   Date rentDate;
   Date dueDate;
   double rentalFee;
   boolean isOverdue() {
      Date now=new Date();
      return dueDate.before(now);
   }
   double getTotalFee() {
      return isOverdue() ? 1.2*rentalFee : rentalFee;
   }
}
class MovieRental {
   String movieTitle;
   int classification;
   Date rentDate;
   Date dueDate;
   double rentalFee;
   boolean isOverdue() {
      Date now=new Date();
      return dueDate.before(now);
   }
   double getTotalFee() {
      return isOverdue() ? Math.max(1.3*rentalFee, rentalFee+20) : rentalFee;
   }
}
```