MTAT.03.094 – Software Engineering

Written Exam – 17 January 2014

Start: 9:15 – End: 11:45

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.

==========

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)

Note: 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 design and develop program code
B: To invent a new programming language
C: To understand the problem domain of a software system to-be-developed
D: To define/evolve the architecture of a software system

Q-02 (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: Aggregation is a specific type of inheritance relationship
C: Composition describes a whole/part relationship
D: A superclass and its subclasses describe a generalization relationship

Q-03 (1 mark): For which of the following requirements engineering activities is the UML notation the least suitable/useful?

Answer choice:
A: Requirements validation (e.g., consistency checking)
B: Requirements analysis
C: Requirements specification
D: Requirements gathering/elicitation

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 not 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 not a test activity
D: Refactoring does 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: Boundary-value testing is a white-box testing technique
C: Black-box testing techniques exploit knowledge about the code that is tested
D: White-box testing techniques exploit knowledge about the code that is tested

Q-08 (1 mark): You compile a program and you see the following compiler message on the screen: 'missing return statement'. Which of the following describes your experience best?

Answer choice:
A: I made an error
B: I detected an error
C: I detected a fault
D: I triggered a failure

Q-09 (1 mark): Which of the following is essential for XP (Extreme Programming) but not for SCRUM?

Answer choice:
A: A role model
B: Pair-Programming
C: Product Owner
D: Sprint Burn-down Chart

Q-10 (1 mark): Which of the following is a suitable unit for effort measurement?

Answer choice:
A: Lines of Code
B: Week
C: Person-Month
D: Duration

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 types of meetings is/are defined by SCRUM?

Answer choice:
A: Product Owner Meeting
B: Sprint Retrospective
C: Weekly Scrum
D: Sprint Review

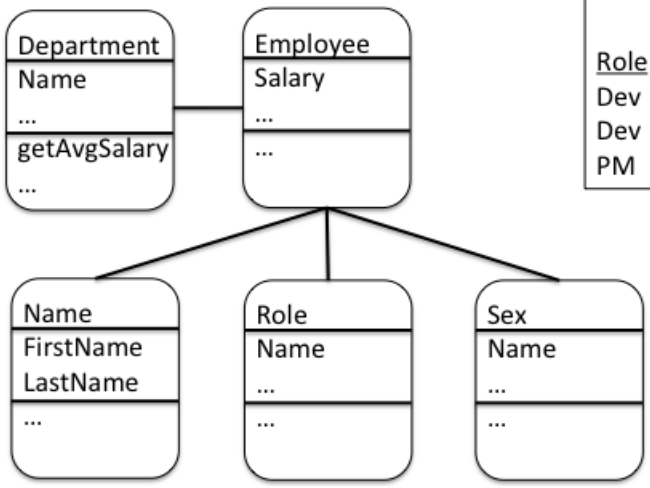
PART 2: Open Questions (10 marks + 3 bonus marks)

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 (point out) the most important differences between the following:
(1) 'Software Engineering' versus 'Programming'
(2) 'Software Engineering' versus 'Software Development in a Craftsmanship-style'
(3) 'Software Engineering' – 'Systems Engineering'

Q-13 (3 marks): First give one example of a 'fault' and one example of a 'failure' (in the context of software development). In you examples, make it clear how the 'fault' and 'failure' you are describing either is introduced/committed, or detected/encountered. Then explain why it is important to distinguish between 'fault' and 'failure'. Ideally, describe benefits from making the distinction and/or describe problems that could emerge if the distinction is not made.

Q-14 (2 marks): In the following excerpt of a domain model, there is (at least) one obvious mistake regarding the conceptualization of the domain classes (i.e., ignore the fact that the exact nature of the relations between classes are not yet fully specified). Spot the mistake and briefly explain why it is a mistake. To get two marks, it is sufficient that you point out one mistake and explain it – even if you think that there is more than one mistake in the domain model.
Note: The excerpt of the domain model below on the left is intended to model (a part of) a payroll system that – among other things – will produce reports similar to the one shown below on the right.



Department: IT
Average salary: 1700

Role	Name	Sex	Salary
Dev	Joe Klein	male	1500
Dev	Jane Connor	female	1600
PM	Karen White	female	2000

Q-15 (2 marks): For each recommendation listed below, state one type of waste that is reduced by implementing the recommendation (Note: the recommendations are made for agile projects):

- Recommendation 1: One person always works on one task at a time.
- Recommendation 2: The items in the product backlog are prioritized (and re-prioritized if needed) and only refined as they move up the priority scale.

Q-16 (3 marks – bonus): Assume, you hear the following sentence: „We decided to measure code maintainability by calculating the complexity of Java methods in terms of MCC (McCabe complexity)." What is the correct mapping of the four underlined terms to the following elements of the described measure: Name (1), Entity (2), Attribute (3), and Unit (4)?

Note: You will get a penalty, if you incorrectly assign an underlined term to any of (1) to (4). For not assigning a term to any of (1) to (4), you won't get a penalty. Also, in case you give more wrong than correct answers, the worst thing that can happen is that you get 0 marks.

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 (2 marks): Assume you have a use case diagram containing 2 Actors (A1, A2) and 5 Use Cases (UC1 to UC5). From the use case descriptions you know the following:

- Actors are of type 'complex'
- UC1 and UC2 have 5 transactions each
- UC3 has 4 transactions
- UC4 and UC5 have 2 transactions each

To Do:
Calculate the Unadjusted Use Case Points (UUCP) for the use case diagram. Show all your calculations and briefly explain each of your steps.

T-02 (6 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) Draw the Control Flow Graph (CFG)
(2) Calculate the Cyclomatic Complexity (McCabe). Show the details of your calculations.
(3) 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: Use the following table to present your test cases:

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

T-03 (2 marks): Two persons independently review the same requirements specification. In total, both reviewers find 10 unique defects (as shown in the table below).

Defect	Reviewer 1	Reviewer 2
D1	1	1
D2	1	0
D3	0	1
D4	1	0
D5	1	1
D6	1	0
D7	0	1
D8	1	0
D9	1	1
D10	1	0

Defect data table:
'0' represents 'not found by reviewer' and '1' represents 'found by reviewer'.

To Do:

Estimate how many defects are still in the document after all defects found by reviewers 1 and 2 have been corrected. (Assume that no new defects are introduced when corrections are made.)

To get full marks you must show all your calculations and explain what estimation (or prediction) model you are using and what assumptions you are making about the nature of the defects.
If you only write down a number for the estimated total number of defects and/or remaining defects in the document, you will get 0 marks (even if the numbers you come up with are plausible).