Computer Science COMS W4156 Advanced Software Engineering Spring 2018 - Midterm Exam Rubric

March 6th, 2018

Do not open the exam until the proctor tells you to do so. You may not use any books or notes. You may not use a calculator or any other device beyond a pen, pencil and eraser. Please write each answer in the corresponding space, continuing on the blank backs of pages if needed. Read through the entire exam before beginning to answer questions. Question 3 is long, with some intermediate pages to provide plenty of space for answers. It is not necessary to use all the space. The exam consists of 12 pages.

Name:

UNI (also put your UNI at **the top of every page**, since the pages will be separated during grading):

Problem No.	**Max Points**	**Points Scored**
1	30	
2	25	
3	20	
Total	75	

Problem 1 – Multiple Choice (2 points for each correct answer; -1 point for each wrong answer or if multiple answers are selected; 0 if blank)

Unless otherwise stated **circle** the letters that represent the best answer. **There may be more than one letter for any question**

1. Draw lines between the methodologies and the most accurate description of that methodology

Methodology
No methodology
Predictive
Agile

Description
Chaotic
Sequential
Adaptive

ANSWER:

No methodology -> Chaotic
Predictive -> Sequential
Agile -> Adaptive

2. Which of the following best describe a **functional** requirements
    a. Quality attributes
    b. Something the system should do
    c. The cost to run the system
    d. The portability of the software
    e. All of the above

ANSWER:

B. Something the system should do

3. Which of the following are **non-functional** requirements
    a. As a Professor I want to be able to establish groups to enable grading group projects

b. An external interface the system must support
c. The ability of the system to scale to 1000 users
d. As a student I want to be able to see the homework
e. All of the above

ANSWER:

B. An external interface the system must support C. The ability of the system to scale to 1000 users

4. Software projects can *fail* by not satisfying:
    a. Functional requirements
    b. Internal Quality Attributes
    c. External Quality Attributes
    d. Constraints
    e. All of the above

ANSWER:

E. All of the above

5. Which of the following statements is **incorrect** with respect to user personas:
    a. A user persona represents a class or type of user of a specific interactive product
    b. A persona encapsulates a distinct set of behavior patterns
    c. Multiple distinct classes of users can be aggregated into a single "average user"
    d. User personas are a tool used to understand the classes of users and therefore their requirements

ANSWER:

C. Multiple distinct classes of users can be aggregated into a single "average user"

6. Which of the below are true in relation to a *single* user story (**circle all those that apply)**

a. A user story can describe a large and complex set of features
b. User stories should be able to be estimated
c. User stories should clearly define the 'what' the user wants to be able to perform
d. User stories should describe the 'why' the user wants this functionality
e. User stories should define architecture or technology selection

ANSWER:

~~A. A user story can describe a large and complex set of features~~
B. User stories should be able to be estimated
C. User stories should clearly define the 'what' the user wants to be able to perform
D. User stories should describe the 'why' the user wants this functionality

7. A product requirement document/product vision specifies **(circle all those that apply)**
   a. Defines the purpose/goal of the product
   b. Specifies the complete project plan with delivery dates that are cast iron commitments
   c. Detailed design and technology choices
   d. Users of the product
   e. Key functional requirements

ANSWER:

A. Defines the purpose/goal of the product
D. Users of the product
E. Key functional requirements

8. With respect to non-trivial programs testing is:
   a. The process of proving software to be correct
   b. The process for revealing defects in software and establishing whether software has reached a degree of quality
   c. The process of exhaustively testing every set of inputs for non-trivial programs
   d. Random selection of inputs to test the program works
   e. All of the above

ANSWER:

B. The process for revealing defects in software and establishing whether software has reached a degree of quality

9. Connect White box and Black Box

Method
White Box
Black Box

Characteristic
Developer is allowed to look at the specification of the code
Developer is allowed to look at the internals

ANSWER:

White Box -> Developer is allowed to look at the internals
Black Box -> Developer is allowed to look at the specification of the code

OR

White Box -> BOTH
Black Box -> Developer is allowed to look at the specification of the code

10. A 'good test case'
    a. Has a high probability of exposing a defect
    b. Has a low probability of exposing a defect
    c. Should test scenarios that are known to work to ensure we do not break the test suite or have to fix the bug
    d. Always passes
    e. None of the above

ANSWER:

A. Has a high probability of exposing a defect

11. Coding conventions (**circle all those that apply)**
    a. Are unnecessary for small teams
    b. Are an important for any codebase to improve readability

c. Are often implemented checked with tool support / linters
d. apply only for the lead engineer on the team
e. None of the above

ANSWER:

B. Are an important for any codebase to improve readability
C. Are often implemented checked with tool support / linters

12. Code smells
    a. Are a problem that must be fixed in a code base
    b. Are bugs
    c. Are indicators that should be investigated but may not always result in a change
    d. Are when the production database overheats
    e. All of the above

ANSWER:

C. Are indicators that should be investigated but may not always result in a change

13. Refactoring is
    a. A software development methodology
    b. Adding features to the code
    c. Dangerous and should never be conducted on a production codebase
    d. Maintaining the functional behavior of the code while cleaning up the internal structure
    e. None of the above

ANSWER:

D. Maintaining the functional behavior of the code while cleaning up the internal structure

14. Which of the following are **not** part of the "Agile Manifesto"
    a. Customer collaboration over contract negotiation
    b. Individuals and interactions over processes and tools
    c. Following a plan over responding to change

d. Working software over comprehensive documentation

ANSWER:

C. Following a plan over responding to change

15. Scrum is
    a. A predictive process used to estimate software deliver dates
    b. A prescriptive checklist defining roles, process and ceremonies to run a software development process
    c. A software development method which is part of Agile Umbrella
    d. A git workflow

ANSWER:

C. A software development method which is part of Agile Umbrella

Problem 2 – Concepts (30 pts)

For the following questions you **do not** have to write prose/paragraphs. Bullet points are acceptable

1. You have been drafted in as a consultant to assist a software engineering team that is in trouble. The team has a good grasp of the requirements and a well defined sprint backlog. However, each month at the end of the sprint the is always late producing a working version of the system to demo and/or release. There team struggles to combine the work of individual developers
    a. Diagnose the problem with the team
    b. Describe the key principles of any practice you would ask the team to put in place

ANSWER:

a) The team clearly has workflow issues; integrating all the features worked on from the backlog/sprint into a single shippable product due to poor coordination and probably little to no code integration by developers right from the start. (we are explicitly looking for CI / they are integration hell)

b) They need continuous integration to improve workflow.

The key principles:
a. Developers work on a local copy of the code and test their changes locally
b. Trigger a continuous integration build as and when new commits are made/checked-in to check if test suite passes
c. Ensuring commits are made regularly so as to avoid code/merge conflicts
d. Make it easy for the team to track build failures and revert them.

RUBRIC:

a) Diagnoses issue with team adeptly: 3points, adequately: 2 points, partially right: 1 point

b) Description of key principles needed by team; adeptly: 3points, adequately: 2 points, partially right: 1 point

2. Mutation testing is a way of measuring the effectiveness of test suites.
   Within mutation testing
   - the ***code under test*** is ***randomly*** mutated/modified to generate a **mutant** version of the code under tests
     - (> are changed to >=, literal values are changed from 5 to 200, etc)
   - The mutations generally changes the ***functional*** behavior of the **mutant** in comparison to the original code under test
   - The test suite is run on the **mutant code**

   Depending on the quality of the test suite what will happen when the test suite is run on the mutant code?

	Test Suite Passes/Fails
Test Suite is Highly Effective	
Test Suite is Partially Effective	
Test Suite is Not Effective	

ANSWER:

	Test Suite Passes/Fails
Test Suite is Highly Effective	**Test suite likely to/will fail on a very large number, if not all tests**
Test Suite is Partially Effective	**Test suite is likely to/will fail on a number of tests**
Test Suite is Not Effective	**Test suite is likely to/will pass**

EL: I will accept a binary interpretation that if at least one test case fails then the test suite is deemed to fail.

	Test Suite Passes/Fails
Test Suite is Highly Effective	**Fails**
Test Suite is Partially Effective	**Fails**
Test Suite is Not Effective	**Passes**

RUBRIC:

Able to correctly identify all expected test suite behaviors: 6 points. Each correct answer is worth 2 points

3. For ‘velocity’ within scrum
    a. Define the meaning of ‘velocity’
    b. How is velocity used within sprint planning?
    c. Can velocity change over time?

ANSWER:

a) Velocity is the rate at which a team is able to complete user story points in a given sprint. It could also me defined as the measure of the number of user story points a team is able to complete in a given sprint. b) Velocity is a critical measure that helps determine the amount of work done, i.e. productivity, and is used to help determine how to plan next set of tasks/sprints ahead therefore, affecting the timelines and roadmap of the deliverables. (velocity is used to estimate the number and complexity of user stories to incorporate into the next sprint) c) Velocity can definitely change over time as it depends on how fast the team is able to pick up tasks and complete them. It’s possible that at the start of development, given the unfamiliarity of tech stack involved, the team is functioning at a slower velocity, but as they gain more expertise, go on to complete them at a much faster pace. Team can also lose skilled team members along the way which can also affect velocity.

RUBRIC:

a) Definition of velocity in terms of agile is clear and shows understanding of concepts: 2 points, definition not sufficient but exhibits some level of understanding: 1 point

b) Clear explanation of the role of velocity in sprint planning: 2 points, partially right answer: 1 point

c) Explanation for why velocity can change over time exhibits understanding of concept: 2 points. 1 point for partial/vague explanation

4. Describe one code smell
    a. What is it?
    b. Why can it indicate a problem?
    c. What is a possible refactoring in response to the code smell?

ANSWER:

a) Too many parameters in a function: This is when a method relies on or has too many input parameters before it can be called/executed.

b) It generally indicates that developer was having a hard time abstracting out the functionality of the software into modular parts or that developer may be thinking about solving the problem in a more complex centralized way. Having many parameters for a function increases the complexity of code and introduces the possibility for many errors is input parameters are not all specified and are in right order and formats/types.

c) Limit the number of parameters you need in a given method, or use an object to combine the parameters. Also, break down function into smaller ones that can call each other. (the refactoring is to create a parameter object)

RUBRIC:

a) Clear example of code smell: 2 points, partially right or vague answer: 1 point

b) Clear explanation about why smell can indicate a problem: 2 points, vague answer: 1 point

c) Possible refactor for smell is clear: 2 points, is partially true: 1 point

5. Order the stages of TDD (assuming the test case fails)

Activity	Step Number
Refactor	
Run the test	
Write code	
Write the test	
Run the test and see if it fails	

ANSWER:

Activity	**Step Number**
Refactor	5
Run the test	4 OR 2
Write code	3
Write the test	1
Run the test and see if it fails	2 OR 4

RUBRIC:

Ordering sin right step: 5 points 1 point for each correct step

Problem 3 – Worked Problems (20 points)

1. Your recommended maximum heart rate is related to your age.

Age	Recommended Maximum Heart Rate
Under 20	200
Under 30	190
Under 40	180

You have been asked to help test a function which takes as input your **current** heart rate and your age. The function returns whether you are exceeding your recommended maximum heart rate (you are exercising too hard)

a. Identify the equivalence partitions (you may draw a graph/diagram)

ANSWER:

(MAX 5 pts)

Looking for a graph of the input space. The x axis is age and y axis is current heart rate.

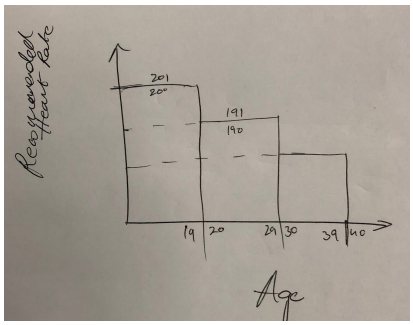
The function to be tested ->

```
def func(age, current_heart_rate):
        SWITCH (valid_age_range):
                If (current_heart_rate > recommended_max_heart_rate):
                        Print ("You're exercising too hard")
                        Return TRUE
                Else:
                        Return FALSE
```

Identify the equivalence partitions (you may draw a graph/diagram) - set of inputs for which we expect the software to behave in the same way (giving an analogous result)

RUBRIC:

Correctly identified partitions encompassing all valid and invalid partitions with correct ranges (shown in a diagram) (5 points); Above with Incorrect ranges (4 points); only one age/heart rate partitions captured (1 point)



b. Using boundary value analysis propose the set of test cases you would execute on the function

Inputs		Output
Age	Current Heart Rate	
19	200,201	
29	190,191	
39	180,181	
20	190,(200,201),191	
30	180,(190,191),181	
40	error	

ANSWER:

(MAX 6 pts)

Errors often occur at the boundaries of ranges: off-by-one error, fencepost error, corner case (e.g., some condition in the code checks < but should check <= or vice versa). The test cases written Need to check at both boundaries of different equivalence partitions stated above.

Each test case should be clearly described with the boundary value analysis clearly stated.

RUBRIC:

6 test cases: (1 point each, partially right or vague answer: 0 point)

2. For the following code identify the ***minimum*** set of test cases that achieve the following set of coverage criteria

```
/*
 * isCarnivore
 */
@staticmethod
def isAnimalDangerous(bool isCarnivore, bool eatsHumans, bool isGrown) {
   bool isDangerous = False
   if (isCarnivore) {
      if (eatsHumans && isGrown){
          isDangerous = True
      }
   }
   return isDangerous;
}
```

a. Statement coverage

Test Case	isCarnivore	eatsHumans	isGrown
1			
2			
3			
4			
5			
6			

ANSWER:

(MAX 3 pts)

1 case: 1,1,1

For the above, 1 = TRUE, 0 = FALSE

RUBRIC:

3 points for correct answer with correct test case coverage, partially right or vague answer (correct number of test cases but wrong values): 1 point

b. Branch coverage

Test Case	isCarnivore	eatsHumans	isGrown
1			
2			
3			
4			
5			
6			

ANSWER:

(MAX 3 pts)

3 cases:
1,1,1
1,0,0 or 1,1,0 or 1,0,1
0, whatever, whatever

For the above, 1 = TRUE, 0 = FALSE

RUBRIC:

3 points for correct answer with correct test case coverage, partially right or vague answer with correct number of test cases: 1 point

c. Condition coverage

Test Case	isCarnivore	eatsHumans	isGrown
1			
2			
3			
4			
5			
6			
7			
8			

(MAX 3 pts)

```
@staticmethod
def isAnimalDangerous(bool isCarnivore, bool eatsHumans, bool isGrown) {
  bool isDangerous = False
  if (isCarnivore) {
    if (eatsHumans && isGrown){
       isDangerous = True
    }
  }
  return isDangerous;
}
```

EL - Within branch coverage we only say has a branch evaluated to T or F (been taken or not). However, there could be latent bugs caused if sub-expressions were not exercised and therefore did not influence the flow of control**.**

- Therefore, condition coverage says 'has each sub-expression evaluated to T and F.'
- In our example our three expressions are isCarnivore, eatsHuman and isGrown
- *Now*, I had an excellent counter that I did not specify the language of the pseudo code.

With **lazy evaluation** if eatsHumans ==F then isGrown will never be evaluated.

isCarnivore	eatsHumans	isGrown
F	Will not evaluate	
T	T	T
T	T	F[x]
T	F	X

[x]to get isGrown to evaluate to False we have to set eatsHumans to T so the second expression is evaluated.

Without lazy evaluation

isCarnivore	eatsHumans	isGrown
F	Will not evaluate	
T	T	F
T	F	T[Y]

[Y] without lazy evaluation even if eatsHumans is set to False then isGrown will evaluate. (this was the original answer)

RUBRIC:

3 points for correct answer with correct test case coverage, partially right or vague answer with correct number of test cases: 1 point