Grading Rubric for COMS W4156 First Exam - October 25, 2016

Problem 1 – Multiple Choice (20 points). 2 points for each correct answer. -1 point for each wrong answer or if multiple answers are selected (or if there is any ambiguity as to which answer is selected). 0 for left blank.

1. d – all of the above
2. d – all of the above
3. b – Class, Responsibility, Collaborators
4. a – Estimate
5. e – none of the above
6. b – Add a new group of tasks to the “To Do” section of the Task Board for the demo preparation and presentation
7. c – Static structure
8. c – Meeting the customer’s needs
9. a – User interface
10. b – How much will it cost and How long will it take

Problem 2 – Vocabulary (30 points). 0 points if blank or totally wrong. 3 points for a correct answer. Use your judgement in awarding partial credit, 1 or 2 points, for answers that aren't really correct but sort of vaguely have some clue, or that say something correct but also say something wrong; try to be consistent across students. Do not grade in half-points, i.e., no 0.5 or 2.5.

a) Pair Programming – Two developers work together on one computer, one typing and the other reviewing.
b) Static Analysis – A tool that examines the source code, without executing the code, for "code smells" and bugs.
c) Burn Down Chart – A graphical representation of work left to do versus time remaining in the iteration. (A picture is not required, but if students do draw a picture, the axes should be labeled to indicate work left vs. time remaining.)
d) Design Sprint – A week spent focusing on a particular problem, where team members sketch potential solutions, develop a "fake it" prototype, and show to customers.
e) Association (in a class diagram) – A generic relationship among classes, meaning interacts with, communicates with, knows about, etc. (A picture is not required, but if students do draw a picture, there should be two or more rectangles with labeled lines between them. Ignore whether or not the lines have arrows.)
f) Aggregation (in a class diagram) – A strong relationship among classes, meaning "has a", whole-part, container-containee, etc. (A picture is not required, but if students do draw a picture, there should be two or more rectangles with labeled lines between them. There should be a diamond or similar marker at one end of each line. Ignore which end and whether or not the diamond is filled.)
g) Non-Functional Requirements – Requirements that do not correspond directly to features, such as reliability and security.
h) Time Box (in agile processes) – Planning what can be completed within a specified time period, such as one calendar month.
i) Single Responsibility Principle (SRP) – Every class (or other language-appropriate program unit) should have a single cohesive functionality and every operation provided by that class should be needed for implementing that functionality.
j) Don't Repeat Yourself (DRY) – Avoid duplication by abstracting out common functionality and putting it in a single place (e.g., class or method).

I cannot think of any reasons to draw pictures except for the questions indicated, but if students do draw other pictures check that they make sense.

Problem 3 – Mini-Project (50 points)

The question parts are open-ended, and there is no single “correct” answer.

Part A Requirements (15 points)

10: Plausible set of user stories for basic task board functionality, similar to github projects (they do not need to follow github’s column format). I’m expecting mostly CRUD operations. They do not need to consider login, authentication, etc.

5: Each story written as a title and a few sentences. It is not necessary to specify a user role if the role would be team member. I’m expecting user stories only for the team members, not the teaching staff, but its fine if they also include teaching staff features.

They do not need to consider the UI or time estimates, but its fine if they do.

Part B Design (15 points)

5: Plausible set of classes that together seem to cover all the user stories from part A, with meaningful names for classes such that you can figure out how they correspond to the features defined by the user stories. For example, there should be at least two classes, corresponding to “task board” and “item”, there may be others.

5: Diagrams for some individual classes, including attributes and operations, with corresponding prose explanations. Meaningful attribute and operation names corresponding to the user stories are sufficient, they do not need to specify initialization, parameters, etc. Do **not** check for proper UML syntax.

5: Diagrams showing some relationships among classes, with corresponding prose explanations. For example, a task board contains items. Do **not** check for proper UML syntax.

They do not need to consider the UI or time estimates, but its fine if they do.

Part C Project Planning (10 points)

Now the students need to consider the UI and time estimates. Note the question is ambiguous as to whether or not the general task board UI was covered by the already-completed first iteration. If the students ask about this during the exam, the response should be that they should make reasonable assumptions - either assuming it was or was not is fine. They do not need to say anything at all about project velocity, which I told them would not be covered by this exam, but if they do ignore it.

5: At minimum, the planning should include (somehow) devising time estimates for the existing user stories, and fitting them into one or more iterations of some predefined length. Note they do not actually have to invent time estimates for the user stories, they just need to explain that they would do so, so ignore any specific time estimates they might include.

5: The planning should also include either expanding the existing user stories or introducing new user stories (or tasks associated with user stories) corresponding to developing the part of the UI that displays items and enables item operations. Ignore anything extra they might include.