UNI:

# Computer Science COMS W4156 Advanced Software Engineering Fall 2017 - Midterm Exam

*October 19, 2017*

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 13 pages, with the last page saying only "(this page intentionally left blank)".

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 | 10 | |
| 2 | 15 | |
| 3 | 35 | |
| **Total** | 60 | |

# Problem 1 – Multiple Choice
# (10 minutes, 10 questions, 1 point for each correct answer)

[1 point if correct, 0 points if incorrect or no answer or multiple answers or ambiguous]

**Circle** the letter that represents the **best** answer to each of the following questions.

1. Which term(s) should be grounds for concern attached to a **user story**?
    a. IETF RFC 2616 ←
    b. Definition of Done
    c. Conditions of Satisfaction
    d. All of the above
    e. None of the above

2. Which of the following is needed for **every** use case?
    a. UML
    b. Authorization and Authentication
    c. A sequence of steps ←
    d. All of the above
    e. None of the above

3. What does **CRC** stand for?
    a. Cyclic Redundancy Check
    b. Class, Responsibility, Collaborator ←
    c. Communication, Resources, Coverage
    d. All of the above
    e. None of the above

4. Which component(s) of a user story should **always** be specified by the customer?
    a. Estimate
    b. Priority ←
    c. Static Analysis
    d. All of the above
    e. None of the above

5. Which of the following should **never** occur during the daily standup meeting?
    a. A demo for a prospective customer ←
    b. A user story is moved to the “Overflow” section of the Task Board
    c. The Burn Down Chart is updated
    d. All of the above
    e. None of the above

6. What should you do when the Customer informs of a **showstopper** bug?
    a. Schedule an All Hands Meeting
    b. Add a new group of tasks to the "To Do" section of the Task Board to try to find and fix the bug, and move all other pending tasks to "Overflow" ←
    c. Add a week to the iteration to address the bug
    d. All of the above
    e. None of the above

7. Which term **best** describes a Class Diagram?
    a. Design Sprint
    b. Model View Controller
    c. Static Structure ←
    d. All of the above
    e. None of the above

8. Which of the following **best** defines "value"?
    a. I know it when I see it
    b. Continuous Integration
    c. Meeting the customer's needs ←
    d. All of the above
    e. None of the above

9. Which of the following is part of the **View** in the MVC architecture?
    a. User display ←
    b. Persistent data store
    c. Microservices
    d. All of the above
    e. None of the above

10. Which of the following does **every** Customer want to know?
    a. What does API stand for
    b. How much will it cost and How long will it take ←
    c. How did you choose your application development framework
    d. All of the above
    e. None of the above

# Problem 2 – Vocabulary
# (15 minutes, 5 questions, 3 points for each correct answer)

Explain the following with prose and/or drawings.

[0-3 points for each part, 0 for no answer or completely incorrect answer, note suggested points below add up to more than 3, but do not give more than 3 points. There may be other correct answers, or partial answers, beyond those listed: try to be consistent on partial credit for partially correct answers.]

a. Project Velocity

A measure of the speed at which the team completes work during each iteration. [1 point]

During iteration planning points (or a similar estimation unit) are assigned for each user story or task (or other work unit). [1 point]

The project velocity is the number of points that the team completed during the most recent iteration (or alternatively the average number of story points completed during several recent iterations). [2 points]

For the first iteration, this should be set to 70% (or some percentage < 100%) of the number of story points the team thinks it can complete during the upcoming iteration, because estimates are imprecise and often optimistic, and various things invariably come up that take time away from the work planned for the iteration. [2 points]

b. Time Box vs. Scope Box

Time boxing is specifying a specific fixed length of time for an iteration, so a team can only complete as many user stories (or other work unit) as will fit into that time period. [2 points]

Scope boxing specifies features (or other work unit) to be completed for the iteration to be finished. [2 points]

Scope and time are proportional such that as scope increases so does time. [1 point]

Agile teams prefer time boxing because it is more predictable and helps control costs. [1 point]

c. Aggregation (in a class diagram)

Aggregation (or open/weak aggregation) corresponds to a HAS-A relationship, drawn with an open diamond at container (or parent) class. [2 points]

Multiplicity may optionally be specified at the containee (child) class, e.g., * for any number, 0..1 for zero or one, 1..N for at least one and at most N. [1 point]

Aggregation denotes shared association, and the lifecycles of the container and containee classes are independent. [1 point]

In contrast, composition (or closed aggregation) corresponds to a CONSISTS-OF relationship, drawn with a closed diamond at container (or parent) class. [1 point]

Composition (closed aggregation) denotes a strong lifecycle relationship, so if a container/parent instance of a class is deleted so are its containee/child instances, and a child may only have one parent. [1 point]

d. Single Responsibility Principle (SRP)

Functionality should be divided into multiple classes such that every class has a single responsibility. [2 points]

Although a CRC card may list multiple “responsibilities”, these should all be services contributing to a single functionality. [1 point.]

SRP violation test: Does it make sense for <class name> to <operation name> itself? [2 points]

Classes should encapsulate one functionality and have only one reason to change. [1 point]

e. Don’t Repeat Yourself (DRY)

Don’t duplicate responsibility or functionality across multiple classes or multiple methods, instead abstract into a single class/method. [2 points]

Don’t copy/paste code from one place in a program to another. [1 point]

Refactor common code that is defined in multiple locations to instead be defined in a single location in the program and used in the other locations. [2 points]

Duplication increases the amount of work required to extend and maintain the software in the future. [1 point]

## Problem 3 – Mini-Project
## (35 minutes, 35 points maximum, 3 questions)

Imagine that you are developing software to manage task boards for agile teams. The primary features that the system needs to provide are:

- An administrator needs to be able to specify who are the members of each team and provide each team with a workspace.
- Team members should be able to make changes only in their own workspace.
- Team members need to be able to set up and manipulate task boards in their workspace.
- Task boards support To Do, In Progress, Completed and Overflow status categories.
- Task boards support CRUD (create, read, update, delete) operations on Items.

The actual questions to answer are on the following pages, parts A, B and C.

## Part A – Requirements (10 minutes, 10 points)

This specification of the project is incomplete. In particular, it does not say much about what is a task board Item, what teams and team members can do with an Item (besides CRUD operations) and/or what the system can do with an Item. Write a set of 3-5 user stories that together specify the *baseline* (highest priority) functionality of team members working with task board Items, **not** including CRUD operations (or synonyms for CRUD operations). Do not be concerned with the system's UI or what the administrator might do with Items. There is no single correct answer.

This question is intentionally open ended, it is impossible to predefine all possible correct answers, but it is important to try to be consistent on awarding credit or partial credit. If in doubt, ask me.

Discussion with the Customer (or Product Owner or similar) as to what Items are intended for, before starting to define these user stories, and/or afterwards getting approval and priority for the team's suggested user stories. [2 points]

User stories should always be of a form similar to "As a <type of user>, I want <feature> so that <goal>" – in particular there should be three parts corresponding to who, what and why. [2 points if all user stories in this form, regardless of the other content of those user stories.]

A use case, with basic flow and alternative flows, is **not** a user story. [Do not give the 2 points for user story format if the student instead gives use cases, but should still be possible to get the other 8 points with use cases or any other plausible format for requirements.]

Including a name or label for each user story is preferred, but optional. [If provided, add 1 point to make up for some other deficiency, if any, but do not give more than 10 points total.]

There should be at least three user stories (or use cases or features or whatever). [2 points each for best three, but no credit for any user story that is just a CRUD operation, including synonyms for CRUD, e.g., Adding an Item to a Task Board is the same as Creating an Item for that Task Board. Max 6 points for this even if someone writes 20 user stories, except for the special case of advanced features, see below.]

Simple features might include: filling in the main content such as a user story or task for each Item; assigning estimates (time or points) to each Item; assigning a team member or pair to each Item; changing the status of an Item, e.g., from To Do to In Progress to Completed or from any status to Overflow. These are all technically "Create or "Update", in the CRUD sense, but are more specific with respect to functionality. Most students will probably provide only these kinds of user stories, that is fine.

Some students may provide more advanced features, which is great but not required for full credit. For example: recording how long in work time an Item really took to go from In Progress to Completed; introducing additional status categories such as Assigned, In Review, In Testing, etc.; instantiating a separate Burndown Chart with the total number of estimate units on day 1 and then updating that Burndown Chart when an Item becomes Completed; calculating Project Velocity at the end of an iteration; introducing a Team Leader or similar (**not** the administrator) with special operations such as being able to reassign a previously assigned Item or change an Item's estimate; managing multiple Task Boards such as moving an Overflow Item from one board to become a To Do item on another board. [Add 2 points if at least one user story is advanced but still well-defined/precise not vague or "too big", which can make up for some other deficiency, but do not give more than 10 points total.]

Continue your answer for part A on this page if necessary (you can also use the backs of pages).

## Part B – Design (15 minutes, 15 points)

Draw a set of class diagrams to implement at least three of your new user stories for Items (part A). Omit CRUD operations. *Explain* your diagrams in prose. Most importantly, discuss *why* you chose this design. Describe any assumptions you made about the design of the other functionality specified as part of the problem setup. Do not be concerned with the system's UI or what administrators do. There is no single correct answer.

An appropriate answer to this question depends heavily on the answer to part A, so again hard to predefine possible correct answers. If some student left A blank or got it mostly/totally wrong and/or otherwise answers B with something that does not seem to match A, show this to me before trying to grade.

There should be two kinds of diagrams, one showing a class with its name, attributes and non-CRUD operations and the other showing associations, aggregations and/or compositions between classes – which might just have names. [1 point for at least one full class showing name/attributes/operations and another 1 point for at least one multi-class diagram showing association, aggregation and/or composition relationships among classes, whether or not these match the new user stories.]

Check that the diagrams collectively cover at least three new non-CRUD user stories from part A. [1 point per user story completely covered, with all classes, attributes, operations and class relationships that seem necessary for that story, up to 3 total.]

Check that the prose that accompanies the diagrams covers the **same** three good non-CRUD user stories covered by the diagrams. It is not necessary to repeat the content of a class diagram in prose, and there may not be anything to say about a self-evident diagram for an individual class, instead the prose should focus on relationships among classes (including any relationships of one instance of a class with other instances of the same class, e.g., nodes in a tree). [1 point per user story whose class diagrams are explained in reasonable prose, up to 3 maximum for this, note this is a different 3 points than above.]

Separate from the design for the new user stories, the answer should state any assumptions made about the design for the functionality specified as part of the problem setup. This might be via prose alone or by inclusion in the set of class diagrams. [2 points if seems reasonable.]

Most importantly, the answer should also include a *rationale* for the design. This might mention specific design principles, cohesion/coupling, MVC, structure imposed by an application development framework, or even design patterns (although the course has not covered design patterns). [Up to 5 points, use your judgment and try to be consistent with partial credit.]

It is not necessary to use generalization/inheritance. If provided, there should be an open arrowhead at the base class/superclass end not the derived class/subclass end. Treat <<interface>>, if present, like generalization but the line should then be dashed not solid. [If provided, and reasonable, add at most 1 point to make up for some other deficiency, if any, but do not give more than 15 points total.]

It is not necessary to include visibility, attribute types and initial values, or operation signatures in individual class diagrams. It is not necessary to include multiplicity for aggregation/composition, or arrows (instead of plain lines) for associations. [If provided, and reasonable, add at most 1 point to make up for some other deficiency, if any, but do not give more than 15 points total.]

Continue your answer for part B on this page if necessary (you can also use the backs of pages).

## Part C – Project Planning (10 minutes, 10 points)

The requirements described in the project setup (except for Items) have all been completed as part of one or more previous iterations, and have already been demo'd to the customer. Describe how you would plan your next iteration, and additional later iterations if necessary, so that your team (of four members) can demonstrate the new user stories for Items to your customer. Now you **do** need to be concerned with the system's UI, and possibly what the administrator does, since you want to show the system to the customer. It will also take your team some period of time to complete the user stories and prepare the demo. Make sure to explain **how** you would plan, i.e., what your team needs to do during the project planning process. There is no single correct answer.

The answers will probably vary widely. There should be some time period prior to or at the beginning of the engineering iteration to plan, but the students do not need to specify how long or exactly when this is. If some student left A blank or got it mostly/totally wrong and/or otherwise answers C with something that does not seem to match A, show this to me before trying to grade. Note suggested points below add up to more than 10, but do not give more than 10 points total.

Planning should include prioritizing the new set of user stories (from Customer or Product Owner), if not already done. [2 points for mentioning user story prioritization as part of planning or stating assumed already done.]

Planning should include estimating time or point units for each user story (from team members or perhaps a Team Leader or Supervisor). [2 points]

Testing should be part of completing each user story, and included in its estimate, but it is ok if there are separate tasks for testing as long as they also have estimates. [2 points for a reasonable discussion of testing, whether as part of user stories or separately.]

Planning should include defining and estimating the tasks required for modifying and testing the system's UI sufficiently to demo Items to the customer. [1 point.]

Planning needs to define/estimate the tasks for preparing a demo script and practicing it. This might or might not include someone taking the administrator role to set up sample task boards to use in the demo. [1 point.]

Since there has been at least one previous iteration, the planning process should take the project velocity from one or more (average) previous iterations into account to determine whether the estimates allow for including all the new user stories (plus demo!) in a single iteration or if multiple iterations will be needed. [2 points for computing project velocity from previous iteration(s) and 2 more points for describing how this is used for planning the upcoming iteration(s), up to 4 points maximum for this topic.]

There might or might not be any explicit connection between parts B and C, e.g., including as part of planning some additional "infrastructure" tasks corresponding to any components of the class diagrams that may not directly correspond to a single specific user story. [If discussed, and reasonable, add no more than 1 point to make up for some other deficiency, if any, but do not give more than 10 points total.]

There might or might not be a "design sprint" prior to the engineering iteration. User stories might or might not be elaborated into use cases or some other requirements format. [If either or both concepts discussed, and reasonable, add no more than 1 point to make up for some other deficiency, if any, but do not give more than 10 points total.]

Continue your answer for part C on this page if necessary (you can also use the backs of pages).

(this page intentionally left blank)