EECS 481 — Software Engineering — Exam #2

- **Write your name and UM uniqname (i.e., email address) on the exam.**
- There are ten (10) pages in this exam (including this one) and six (6) questions, each with multiple parts. Some questions span multiple pages. If you get stuck on a question, move on and come back to it later.
- You have 1 hour and 20 minutes to work on the exam.
- The exam is closed book, but you may refer to your two page-sides of notes.
- Even vaguely looking at a cellphone or similar device (e.g., tablet computer) during this exam **is cheating**.
- Please write your answers in the space provided on the exam. Clearly mark your solutions. You may use the backs of the exam pages as scratch paper. Do not use any additional scratch paper.
- Solutions will be graded on correctness and clarity. Each problem has a relatively simple and straightforward solution. We may deduct points if your solution is far more complicated than necessary.
  - *Good Writing Example:* Testing is an expensive activity associated with software maintenance.
  - *Bad Writing Example:* Im in ur class, @cing ur t3stz!1!
- If you leave a non-extra-credit portion of the exam blank, **you will receive one-third of the points for that small portion (rounded down) for not wasting time.**

UM uniqname: _____

NAME (print): _____

UM uniqname: (yes, again!) _____

Problem	Max points	Points
1 — Delta Debugging	15	
2 — Requirements Elicitation	15	
3 — Design Patterns	14	
4 — Design for Maintainability	20	
5 — Interviews	16	
6 — Other Topics	20	
Extra Credit	0	
TOTAL	100	

How do you think you did? _____

1 Delta Debugging

Consider applying the Delta Debugging algorithm to the task of skill-based personnel assignment for a development project. A finite set of skills $S = \{s_1, \ldots, s_m\}$ is necessary to complete the project. For each skill s_i, the project must be assigned at least one developer who possesses that skill s_i. There is a finite set of developers $D = \{d_1, \ldots, d_n\}$ available, and each developer has an individual set of skills given by $\mathsf{skills} : D \to S$. For example, it could be that $\mathsf{skills}(d_2) = \{s_1, s_3\}$ while $\mathsf{skills}(d_5) = \{s_3, s_6, s_8\}$. Collectively, the full set of developers has all of the necessary skills S. You are interested in finding a smaller subset of developers that also has all of the necessary skills. Formally, a candidate set of developers D' is $\mathsf{interesting}$ if the union of all skills held by everyone in D' is equal to S.

(2 pts.) In general, this problem formulation violates at least one of the fundamental assumptions of the basic Delta Debugging algorithm. Identify the most important such unmet assumption.

(10 pts.) Provide a simple example that shows that your chosen assumption is violated. You must do so with $n = 3$ by giving definitions for S, D and skills.

(3 pts.) We believe the best possible running time to find a minimal subset of an arbitrary set with an arbitrary deterministic interesting function is $\mathcal{O}(2^N)$. Delta Debugging advertises a better running time. Explain this apparent contradiction between Delta Debugging's running time and more general theoretical bounds. Use at most three sentences.

2 Requirements Elicitation

(4 pts.) Given an example of a *conflict* that might arise during requirements elicitation. Indicate a best practice to *resolve* that conflict.

(6 pts.) Consider the following claim: "*Validation* is less expensive then *verification*, but mistakes made during *validation* are more expensive than mistakes made during *verification*." In about three sentences, support or refute this claim.

(5 pts.) At the end of his lecture, Jason Mars described a situation in which Clinc made a significant mistake in *stakeholder analysis*. All but two of "decision making", "different needs", "exploring alternatives", "organizational position", "personal objectives", and "traceability" were relevant in that anecdote. In at most five sentences, briefly summarize the situation and indicate the four relevant aspects.

3 Design Patterns

Consider the following *incorrect* code for implementing a *Singleton* design pattern, adapted from James Perretta's lecture.

```
1  class Singleton:
2    @staticmethod
3    def get():
4      return Singleton._instance
5
6    _instance = None
7
8    def __init__(self):
9      if Singleton._instance is None:
10       Singleton._instance = Singleton()
11     self._state = 42
12
13   def current_state(self):
14     return self._state
15
16 def main():
17   print(Singleton.get().current_state())
```

(10 pts.) In at most four sentences, indicate the defect in this code and how you would fix it. Be specific.

(4 pts.) In the *Model-View-Controller* design pattern, two pairs of components typically communicate (i.e., depend on each other, call methods from each other, etc.) but one pair does not. Identify the components that should *not* communicate. Note that communication here is not *commutative* (for example, Alice could communicate with or depend on Bob, while at the same time Bob might not communicate with or depend on Alice). In at most three sentences, indicate some ways in which subsequent maintenance would be complicated if that pair were, mistakenly, to be tightly coupled.

4 Design for Maintainability

(4 pts.) Following Wikipedia, "*Scalability* is the capability of a system, network, or process to handle a growing amount of work, or its potential to be enlarged to accommodate that growth." In at most three sentences, support or refute the claim that scalability is a functional property. You will receive +1 bonus point if you *substantially* (in our opinion) relate your answer to Adam Brady's lecture on Google.

(8 pts.) You are considering developing a *multi-language* project. Two requirements include the *performance* of the system (i.e., how fast it runs) and also how the *debugging cost* of the system (i.e., how fast defects can be repaired). For each requirement, list both an advantage and a disadvantage of a multi-language design. Use at most four sentences (one for each pairwise consideration).

(8 pts.) Explain the relationship between a *verifiable* quality requirements and *measurement*. Highlight at least one *risk* associated with measurement *uncertainty*. Use at most four sentences.

5 Interviews

You are responsible for *giving* a non-behavioral technical interview to job candidates; you are the interviewer. Your company views technical interviews as an assessment of software engineering skills. The programming problem you ask of candidates is:

> Two strings are said to be anagrams of one another if you can turn the first string into the second by rearranging its letters. For example, "table" and "bleat" are anagrams, as are "tear" and "rate". Your job is to write a function that takes in two strings as input and determines whether they're anagrams of one another.

The candidate's complete response is below. The first two commented lines indicate questions the candidate asked you.

```
1  /* Q: Do I need to worry about upper- and lower-case? A: No. */
2  /* Q: Will the strings always be the same length? A: No. */
3
4  private boolean areAnagrams (String first, String second) {
5    return areAnagramsRec ("", first, second);
6  }
7
8  /* helper function: this is slow but correct */
9  private boolean areAnagramsRec (String soFar, String remaining, String target) {
10   if (remaining.length () == 0) {
11     return soFar.equals (target);
12   }
13   for (int i = 0; i < remaining.length (); i++) {
14     String whatsLeft = remaining.substring (0, i) +
15       remaining.substring (i + 1);
16     if (areAnagramsRec (soFar + remaining.charAt (i), whatsLeft, target))
17       return true;
18   }
19   return false;
20 }
21
22 /* test 1: "able", "bale"
23    test 2: "astronomer", "moonstarer" */
```

(2 pts. each) Identify two things that the candidate did well.

(3 pts. each) Identify and justify four significant things that the candidate did poorly.

6 Other Topics

(3 pts.) You design a neural representation (or "mind reading") experiment to determine if patterns of neural activation in the brain are similar for "talking about code" and "talking about prose". You use an fMRI or fNIRS device to measure the blood oxygen level dependent (BOLD) signal. You randomly sample, from the Mozilla Firefox project, methods ranging in size from 10–20 lines. While measured by the device, participants read the methods and are summarize them in their own words. In at most three sentences, explain the most significant reasons why it will be difficult to use this experimental setup to answer this research question, and also why it may be difficult to use the BOLD signal to assess software engineering.

(5 pts.) Consider a hypothetical *pair test generation* activity in which one developer constructs test *inputs* while another developer constructs test *oracles*. Support or refute the claim that this activity will reap similar benefits for testing as pair programming does for programming. Use at most four sentences.

(3 pts.) Explain the relationship between *anti-patterns*, *static analysis* and semi-automated *refactoring*. Use at most three sentences.

(4 pts.) Describe a specific situation (or sketch a small method) in which a static analysis tool (such as Infer or CodeSonar) would issue a false alarm and indicate the alarm type. Then describe a situation (or sketch a small method) in which such a tool would have a false negative and indicate the defect type.

(5 pts.) A senior software engineer who leads your team at a large software company suggests that the team start a paired activity. In this activity, pairs email each other code and changes and agree to merge files together. Explain why this activity is pair programming, pass-around code review, both, or neither. Support or refute the claim that this paired activity would be effective at improving code *readability*. Use at most five sentences.

7 Extra Credit

What is one thing you would tell future students who are considering taking this class?

In retrospect, what is one thing you enjoyed about this class?

In retrospect, what should be changed about this class for next year?

From Beck et al.'s *Industrial Experience with Design Patterns*, list one of the "lessons learned".

In Haraldsson et al.'s *Fixing Bugs in Your Sleep: How Genetic Improvement Became an Overnight Success*, what did the system do?

From Chi et al.'s *Expertise in Problem Solving*, list one way in which experts and novices "chunk" problems differently.