EECS 481 - Exam #2 - ***KEY***

Winter 2020 - Software Engineering

Instructions (read carefully, please!)

1. Please obtain your own editable copy via Google Docs (requires UM login):
    a. https://docs.google.com/document/d/1XVFoPFWCLwzEEbT1qhR86TkjvZPl74nDChLWC4yBQHk/copy
2. The expected time for this take-home exam is two hours.
    a. Expected time-and-a-half for the exam is three hours (e.g., for SSD).
3. The exam is **optional**. If you do not submit a valid exam file by the deadline, or if your last submission has “do not grade” instead of your name, you are skipping the exam.
    a. If you elect to skip the exam, your course grade will be out of 82 instead of out of 100. See the course forum for details.
    b. If you do not elect to skip the exam, we will mark your exam and it will count for 18% of your grade.
4. The exam file is due **Thursday, April 16th at 11:59pm Eastern** via Gradescope.
    a. You may resubmit your exam file as often as you like. We will grade the last submission file (unless you mark it “do not grade”, in which case we will not grade your exam at all). Your PDF must be 12 pages matching this layout.
5. You must use a word processor to **type your answers** by editing the exam file. You may not hand write the exam and scan it in.
6. You must type your answers only in the **framed answer boxes**.
7. You must **not change the size** or position or margins of the framed answer boxes.
8. You must use the same **11 point Arial** typeface for all of your answers. If your answer for a question does not fit in its provided box, edit and simplify your answer.
9. If you leave a non-extra-credit answer box **blank or type “skip”** in it, you will receive one-third of the points for that subquestion.
    a. Because the exam is optional and you can skip questions, grading on answers you do include will be **quite strict** compared to other assignments.
10. The exam is **open book**, open notes, open computer, and open Internet.
11. You must **work alone** to complete the exam. You may quote or refer to text from the readings or Stack Overflow, for example, but you must turn in your own work and may not collaborate with other humans.
12. On certain questions you will be required to craft examples. You will **not receive credit** for class-, reading- or Internet-derived examples. In other words, examples must be personal in some aspect (whether real, figurative or imaginative).
    a. Coincidental overlap with other students will be investigated manually.
13. We will use prose **plagiarism detection** software and investigate reports manually.
14. Please use the **one stickied Piazzas forum thread** for all public exam clarifications.
    a. As with in-person exams, we can’t say much beyond strict clarifications.

Q1. Logistics and Extra Credit (3 points)

Q1a (1 pt.). If you wish your exam to be graded and you understand the instructions from the previous page, type your name in the box below. If you previously submitted an exam file but no longer wish your exam to be graded, type “do not grade” in the box below and re-submit.

ANSWER KEY // (This answer key focuses on the technical content rather than creative SE situations.)

Q1b (2 pts.). Type an “X” to the left of every claim that is true about this exam. (Because this is a remote exam and many students are concerned about exam integrity, the instructions for this exam are quite specific. Refer to the Instructions above.)

	I can resize answer boxes on this exam to make my answers fit.
X	I can use material from the Internet on this exam.
	I can change the font on this exam to make my answers fit.
	I can write examples from the lectures for short answer questions on this exam.
X	This exam will be graded more strictly than previous assignments.

Q1c (2 pts extra credit). Suppose or refute the claim that Delta Debugging could be used to replace the input minimization algorithm in “Minotaur: Adapting Software Testing Techniques for Hardware Errors”.

Very likely “support”. Minotaur measures hardware tests in terms of “resiliency” = “how many instructions are visited“ = “coverage” and seeks tests that have the same coverage but are smaller and faster. Minotaur already uses a “a simple, greedy algorithm based on binary search for the Minimizer.” Since Delta Debugging is also a heuristic algorithm based on binary search, it is likely applicable. However, a full discussion should cover Delta Debugging’s assumptions. Notably, this problem is not unambiguous: if {X,Y} has good coverage and {A,B} has good coverage, their intersection need not. This can be heuristically mitigated (e.g., for saving each “best champion minimized set so far” and double-checking each one at the end to see which one actually retains coverage). However, students who choose “refute” would likely argue against the use of Delta Debugging here based on the violation of that assumption.

Q1d (2 pts extra credit). Argue in favor of one recommended guideline from "Hiring is Broken: What Do Developers Say About Technical Interviews?" and argue against another.

"Use rudimentary questions for screening", "Share the interview description in advance", "Offer alternative interview formats", "Use a real problem", "Solve problems as colleagues, not as examiners."

Typically arguing for one guideline involves just quoting the paper. Students could argue against certain guidelines, like "offer alternate interview formats", in very specific situations. For example, if there is a very specific action that must be undertaken (e.g., "land this flying airplane") it may be that you want to test exactly that action and not offer candidates the possibility of sketching it on paper. Pragmatically, you could argue against "solve problems with colleagues" on cost grounds (those colleagues are now spending time in the interview process).

Q1e (2 pts extra credit). Identify a single case study company from Beck et al.'s "Industrial Experience with Design Patterns" that you think is most indicative of modern practice and support that claim.

Students have free choice here; any well-supported argument would work. A popular choice might be Section 2.3 ("Motorola's Cellular Infrastructure Group" + "Design Patterns"). Cell phone infrastructure is still timely, but more importantly, issues like coupling metrics and ambiguous specifications are still concerns in more modern times.

Q1f (2 pts extra credit). What is one thing you liked about this class? What is one thing you would change for next semester?

:-)
:-(

Q2. Delta Debugging (22 points)

Q2a (10 pts.). Consider Delta Debugging (as defined on Slide 42 from the lecture) applied to the set {0, 1, …, 9}. Suppose Delta Debugging queries Interesting() on the following sets, in order:

```
{0, 1, 2, 3, 4}          // no
{5, 6, 7, 8, 9}          // no, so interference
{0, 1, 5, 6, 7, 8, 9}    // no
{2, 3, 4, 5, 6, 7, 8, 9} // no, so interference again!
{0, 2, 3, 4, 5, 6, 7, 8, 9} // no
{1, 2, 3, 4, 5, 6, 7, 8, 9} // yes
{0, 1, 2, 5, 6, 7, 8, 9} // no
{0, 1, 3, 4, 5, 6, 7, 8, 9} // no, so interference again!
{0, 1, 2, 3, 5, 6, 7, 8, 9} // yes
{0, 1, 2, 3, 4, 5, 6}    // no
{0, 1, 2, 3, 4, 7, 8, 9} // yes
{0, 1, 2, 3, 4, 7}       // no
{0, 1, 2, 3, 4, 8, 9}    // no, so interference again!
{0, 1, 2, 3, 4, 7, 8}    // yes
```

Assume Interesting() is monotonic, unambiguous and consistent. What was the final result (i.e., the returned one-minimal subset) from that run of Delta Debugging?

{1,2,3,7,8}

Interesting(X) is true of any superset of {1,2,3,7,8}. See // comments above in trace.

Q2b (4 pts.). Consider Delta Debugging (as above) with Interesting(X) returning true iff the sum of the numbers in X is greater than 9. What will Delta Debugging return on the set {0, 1, …, 7}?

{6,7}

In this particular case, this answer is true regardless of whether you go “left” first or “right” first on splits.

```
{0,1,2,3}  // no
{4,5,6,7}  // yes
{4,5} // no
{6,7} // yes
{6} // no
{7} // no, so interferences
```

Final answer is {6,7}

Q2c (5 pts.). Type X left of each claim that is true for the "greater than 9" situation from Q2b:

X	Interesting() is monotonic.
	Interesting() is unambiguous. // {4,5,6} is interesting, {6,7} is interesting, but not {6}
X	Interesting() is consistent.
X	Delta Debugging returned a one-minimal subset. // {6} and {7} are uninteresting

X	Delta Debugging returned a minimal subset.
	// There is no subset of size one in {0,1,...,7} that sums to more than 9.
	// So "sets of size two" are minimal answers for this query. DD returned a
	// set of size two, so it returned a minimal subset.

Q2d (3 pts.). Consider Delta Debugging on {0, 1, 2, 3}. Consider a definition for Interesting(X) such that:

1. There is a set P = {1, 3} such that Interesting(P) is true but P is not one-minimal.
2. There is a different set Q such that Interesting(Q) is true and Q is one-minimal.
3. There is a different set R that is one-minimal but not minimal and Interesting(R) is true.
4. Delta Debugging (from Slide 42 of the lecture) on {0, 1, 2, 3} returns R.

Give a definition for Interesting(X) that accepts the smallest number of sets possible such that all four properties above are satisfied.

# P = {1,3}, Q = {1}, R = {2,3}
Interesting(X) = (X = P) or (X = Q) or (X = R)

1. Interesting(P) is true, but P is not one-minimal (because you can remove {3} from it and get {1} which is also interesting).
2. Interesting(Q) is true and Q is one-minimal (because if you remove anything from it you get {} which is not interesting).
3. Interesting(R) is true. R is one-minimal, because both {2} and {3} alone are uninteresting. However, R is not one-minimal, because an answer of size 1 exists: |Q|=1 and Interesting(Q)=true.

Q3. Localization and Profiling (14 points)

Q3a (8 pts.). Create a small Python procedure harrold(x,y) with the following properties:

1. It accepts two integer arguments x and y.
2. It invokes a "divide(_, _)" function that fails when the second argument is zero.
3. It has another line that is simply "safe_function() # false alarm" to mark a false alarm.
4. The positive test {x=1,y=2} does not trigger the bug (i.e., does not divide by zero).
5. The negative tests {x=3,y=4} and {x=5, y=6} do trigger the bug (i.e., divide by zero).
6. A fault localizer, such as Tarantula or Ochiai, ranks the false alarm line as more suspicious than the line with the real bug on the tests {x=1,y=2} and {x=3,y=4} {x=5,y=6}.
7. Your example cannot be taken from class or the readings or the Internet and should be distinct from other student examples if possible.

```
1: def harrold(x,y):
2:   if (x == 3) or (x == 5):
3:     safe_function() # false alarm
4:     x = 0
5:   divide(y, x)
```

Test T1 {x=1,y=2} does not divide by zero. It visits lines {1,2,5}. Passed test.
Test T2 {x=3,y=4} does divide by zero. It visits lines {1,2,3,4,5}. Failed test.
Test T3 {x=5,y=6} does divide by zero. It visits lines {1,2,3,5,5}. Failed test.

Ochiai(S) = failed(s) / sqrt(totalfailed * (failed(s) + passed(s)))

Ochiai(Line 3) = 2 / sqrt(2 * (2 + 0)) = 1.00
Ochiai(Line 5) = 2 / sqrt(2 * (2 + 1)) = 0.82

Ochiai ranks the false alarm line above the line with the real bug.

The "trick" here is to arrange for the false alarm line to be visited only on negative runs but for the real bug line to be visited on every run.

Q3b (6 pts.). You are the interviewer at a company. Type an interview question that centers around profiling. The question must include one opportunity for the applicant to fail to demonstrate technical mastery and also one opportunity for the applicant to fail a behavioral aspect. You must provide a hypothetical situation or framing for the question that is not from the class or the readings or the Internet and is distinct from other student answers if possible.

Many answers are possible.

A technical mastery aspect of profiling might relate to, for example, sampling-based profiling missing periodic behavior.

A behavioral aspect might relate to, for example, a time when you had a conflict with a teammate.

The framing could be anything. For example, you work for a company that makes software for butterflies.

Putting it all together:

“Tell me about a time you and a co-worker disagreed over the profile results for an activity that you know often happens regularly (e.g., the wings flapping) and how you resolve that conflict.”

Q4. Design Patterns & Maintainability (14 points)

Consider the following three maintenance design goals (A-C) and three design patterns (X-Z).

A. Design for Code Comprehension	X. Observer Pattern
B. Design for Change Documentation	Y. Singleton Pattern
C. Design for Testability	Z. Template Method Pattern

In all parts of this question, any examples you devise may not refer to course or Internet material and must be distinct from other student examples if possible.

Q4a (7 pts.). Choose one design goal (A-C) and one pattern (X-Z). Argue that the pattern specifically supports the design goal. Call out at least two properties of the pattern and two aspects of the goal. Devise a brief example setting in which the pattern supports the goal.

Many answers are possible. Consider B-X. In class we argued that not using the Observer pattern leads to the publisher and subscriber being tightly coupled, so changes to one must be reflected in the other. That also applies to change documentation: a patch to such a system using the Observer pattern that only changes publisher-related behavior need only document publisher-related changes (unlike an anti-pattern system, where the change would touch both and thus the change documentation would touch both). You could also argue that it is easier to write “Why” documentation for Observer pattern changes, rather than focusing on the “What” of how the publisher and subscribed are coupled.

Many example settings are possible.

Q4b (7 pts.). Choose a different design goal (A-C) and a different pattern (X-Z). Argue that there is at least one example instance in which the pattern, even when applied correctly, specifically hinders the goal. Be concrete about the metric or fashion by which the goal is hindered.

Many answers are possible. For example, you could pick C-Y and argue that the Singleton pattern complicates design for testability. Even when implemented correctly, the Singleton still encapsulates global state, so all of the testing problems associated with global variables apply. In addition, we described in class the Singleton.get_instance() problem (almost like a “race condition”). Testing whether your code correctly handles different concrete copies of the same logical item may be tricky: a special test harness that forces multiple concrete copies with different values may be required. Even if there is high coverage, the values may or may not show the bug: thus the metric or issue at hand is closer to “traceability to requirements” than pure “line coverage”.

Q5. Requirements and Elicitation (15 points)

Q5a (6 pts.). Consider a requirement elicitation and software development situation involving

1. Requirements involving a quality property of interest are incorrectly captured
2. Stakeholders do not believe the system satisfies that property
3. That system failure is caught by testing but not by static analysis
4. The resolution involves refining the requirements document and the source code

Describe and detail a situation that satisfies the above criteria. You may not use an example from course material or the Internet, and must be distinct from other students if possible.

Many answers are possible. The real constraint is (3): the problem has to be possible/likely to catch with testing but not with static analysis. A direct quality property that can be assessed by testing is “runtime” (performance): you run the program and see how long it takes. However, you cannot decide how many times a loop will execute via a static analysis in general (halting problem) so you cannot predict program runtimes in advance via static analysis in general. “Ambiguity” (as in slide 45 of https://web.eecs.umich.edu/~weimerw/481/lectures/se-13-req.pdf) is a direct RE mistake to pick here. Something like “other modules shall be terminated as soon as the network file transfer occurs”. If stakeholder meant “other modules shall be terminated just before the network transfer completes” but the code implemented “other modules shall be terminated by the time the network transfer finishes”, then the resolution involves both refining the requirements documentation and also changing the source code to make it faster.

Q5b (9 pts.). Your software company plans to outsource its technical interview activities to a human resources subcontractor that specializes in assessing candidates. The subcontractor will inspect many applicants and identify those that meet your needs. The subcontractor wants a requirements document describing desired candidates. Identify one quality property and one functional property associated with modern skills-based interviews. For each, identify an error or mistake in a hypothetical RE (Requirements and Elicitation) conversation with the subcontractor and how you would correct it. Finally, phrase each as correctly as you can using terminology and best practices from RE. You may not use an example from course material or the Internet, and must be distinct from other students if possible.

Quality Property	Many examples are possible … Availability (when interviews can be conducted)
RE Mistake & Correction	Structure clash. Following the course slide example (which you'd have to reword here), the RE documentation might say both "interviews can take place Friday before 5pm" and also "interviews can take place Friday". The hypothetical conversation would be about when on Friday an interview can start or take place (e.g., what if it starts at 4:59pm on Friday?).
Correct Requirement	Resolve weak conflict by clarifying boundary conditions. "Interviews may begin Mondays through Fridays, 9am to 4pm and must last no more than one hour each."

Functional Property	Many examples are possible ... Interviews must identify candidates who have mastered "line coverage"
RE Mistake & Correction	Designation clash. "People involved in the interview must be able to correctly answer questions about line coverage." It's not clear if "people involved in the interview" are the job candidates or the people conducting the interview. Hypothetical question: "Are you saying that our interviewers have to have a certain mastery of the material to assess job applicants, or are you saying that you only want us to recommend job candidates who have mastered the material?"
Correct Requirement	"Job candidates must be able to correctly answer 9 out of 10 questions from the standard Line Coverage Question Bank, chosen at random, within 5 minutes each per question."

Q6. Expertise and Productivity (18 points)

Q6a (8 pts.). In Chi et al.'s "Expertise in Problem Solving", a claim was made about how experts and novices cluster or categorize problems differently. In class and in the reading, physics problems were considered. You will consider test suite quality metric problems instead. Briefly describe four example problems (A, B, C and D) associated with test suite quality metrics. Your examples must not be isomorphic to each other or inverses of each other (e.g., you cannot base one on line coverage and another on lines not visited). Then explain why novices would cluster {A,B} together and {C,D} together, while experts would cluster {A,C} together and {B,D} together. Your explanations should highlight your mastery of both expertise and the nuances of test suite quality. You may not use examples from course material or the Internet (although you may, and likely will, use metrics from the course material, etc.), and must be distinct from other students if possible.

A	Many answers are possible … Compute line coverage of this program with while loops
B	Compute the mutation score of this program with while loops
C	Compute the branch coverage of this program with switch/case statements
D	Find all function calls in this program with switch/case statements
Why Novices Cluster {A,B} and {C,D}	In "Expertise in Problem Solving", novices clustered problems based on gross surface feature similarity. Both A and B involve while loops, while C and D involve switch/case.
Why Experts Cluster {A,C} and {B,D}	Experts clustered problems based on how you would solve them under the hood. Both A and C involve instrumenting the program and printing out when various events are reached, then running the result and counting. Both B and D involve analyzing the AST to look for specific patterns (e.g., "find all X<Y comparisons and negate them" might be a possible mutation operator in B, while "find all function calls and print out each one" would work for D).

Q6b (5 pts.). In Huang et al.'s "Distilling Neural Representations of Data Structure Manipulation using fMRI and fNIRS", a paper not assigned for this class, the following result is reported: "The brain works measurably harder for more difficult [as measured by Big-Oh notation] software engineering problems (in terms of cognitive load). Moreover, the regions activated suggest a greater need for effortful, top-down cognitive control when completing challenging [array and list data structure] manipulation tasks." Support or refute the claim that this result is aligned with the findings in Siegmund et al.'s "Measuring Neural Efficiency of Program Comprehension".

Both support and refute are possible. While both papers mention cognitive load, the former involves showing students graphical data structure problems and finds, informally, that more nodes = more work for your brain. The latter finds that beacons ease comprehension. You could argue, on the support side, that beacons give you fewer conceptual chunks to think about, thus reducing the "informal Big-Oh difficulty" of code comprehension, and thus they are aligned. You could argue, on the refute side, that one of the results is about reading code and one of the results is about doing pencil-and-paper tree problems (no code), so they are not even really talking about the same thing and are thus not aligned.

Q6c (5 pts.). In "The Costs and Benefits of Pair Programming", by Cockburn and Williams, the following example is considered: "A team leader given four junior designers to design a graphics workstation, was also given a private office. After a few weeks, he felt uncomfortable with the distance to his team, and moved his desk to the floor with the other designers. Although the distractions were great and his main focus was not teaching the other designers, he was able to discuss with them on a timely and casual basis. They became more capable, eventually reducing the time he had to spend with them and giving them skills for their next project." Briefly describe an example software development situation (not from class, distinct from others, etc.). Then, with respect to that situation, support or refute the claim that pair programming or mentoring is effective process investment in maintainability.

Many answers are possible. A key here is the notion of a software process investment in maintainability: that requires an analysis of bost costs and benefits. The costs are usually opportunity costs: time or personnel allocated to this activity could instead be allocated to some other baseline activity (e.g., simple testing or just writing documentation, etc.). So a key part of the argument would be claiming that pair programming or mentoring obtains some value for maintainability above and beyond other options.

A support argument would likely build on the evidence (from the lectures and papers) that pair programming produces smaller code and fewer defects. Smaller code takes less time in code inspection and avoiding defects early is a significant investment win. The "example math slide" in the lecture does just this.

You could argue refute (e.g., no evidence is available on mentoring helping maintainability), but support is likely a bit more direct given the material covered in class.

Q7. Language and Repair (14 points)

Q7a (8 pts.). Consider the Eraser analysis (from Savage et al.'s "Eraser: A Dynamic Data Race Detector for Multithreaded Programs"). It may or may not report a race condition and a real race condition may or may not actually be present. For each such combination, describe a situation involving a multi-language Java-and-C project that would result in that outcome. In each case, whether or not the report is made or not and whether it is correct or not should be described in terms of the multi-language aspect, not on other aspects of dynamic analyses (e.g., assume a high quality test suite, etc.). You should not use course examples or examples from the Internet directly, and must be distinct from other students if possible.

True Positive	Consider a Java-and-C project where Eraser can "see" both the Java locks and the C locks and it finds that a variable accessed from both the Java side and the C side has no consistent lock set: a real race condition (true positive).
False Positive	Consider a Java-and-C project where Eraser can only "see" the Java code and does not understand that a C variable is being used as a lock or mutex. Eraser will think a shared variable is being accessed with no locks because it cannot "see" the C lock.
True Negative	Consider a Java-and-C project where Erase can "see" both the Java locks and the C locks and a shared variable is accessed correctly by both C and Java code with a consistent lockset. No race condition, no report: true negative.
False Negative	This one is a bit tougher. Google will reveal some corner cases (e.g., thread initialization code). However, this is fairly easy in a multi-language setting: we just need eraser to think every memory access to a shared variable is guarded by a lock. Suppose Eraser can "see" memory accesses in Java but not in C (e.g., because it instruments Bytecode). If the variable is guarded by a lock in Java but not in C, every access Eraser "sees" will look fine, but there will really be a race condition.

Q7b (6 pts). Support or refute the claim that “it is a better business decision, with respect to repair cost and repair quality, to use automated program repair in conjunction with automated test input generation than it is to use automated program repair alone”. Your argument should speak to these concepts and include an example situation that you devise. Your example should not come from class or the Internet and must be distinct from other students if possible.

This is an active research area, so multiple answers are possible. GIven the context in this class, however, “refute” is the most likely. A key challenge in automated program repair is test suite quality: high-quality test cases (inputs and oracles) are necessary to ensure that the generated repair actually reflects stakeholder requirements (cf. traceability) rather than just cheating its way past the test cases. Automated test input generation (note: the question talks about input generation, not oracle generation!) does not provide any oracles, so it does not provide full test cases: additional high-coverage inputs with no way to tell if the behavior is right or wrong will not help guide automated program repair to produce high quality patches.

Multiple example situations are possible, but for any setting, you could describe a situation in which there is an error-inducing input, and test input generation even produces that input, but since there are no oracles, a low-quality patch is still produced.