University of Illinois at Urbana-Champaign
Department of Computer Science

Final Exam

CS 427 Software Engineering I
Fall 2012

December 17, 2012

TIME LIMIT = 3 hours
COVER PAGE + 12 PAGES

Upon receiving your exam, print your name and netid neatly in the space provided below; print your netid in the upper right corner of every page.

Name: _____
Netid: _____

This is a closed book, closed notes examination. You may not use calculators or any other electronic devices. Any sort of cheating on the examination will result in a zero grade.

We cannot give any clarifications about the exam questions during the test. If you are unsure of the meaning of a specific question, write down your assumptions and proceed to answer the question on that basis.

Do all the problems in this booklet. Do your work inside this booklet, using the back of pages if needed. The problems are of varying degrees of difficulty so please pace yourself carefully, and answer the questions in the order which best suits you. Answers to essay-type questions should be as brief as possible. If the grader cannot understand your handwriting you may get 0 points.

There are 8 questions on this exam and the maximum grade on this exam is **100 points**.

Page	Points	Score
1	12	
2	14	
3	8	
4	4	
5	8	
6	8	
7	13	
8	5	
9	6	
10	6	
11	10	
12	6	
Total:	100	

1. Reverse Engineering

[8] (a) After you have worked at a company for a year, you are asked to make some changes to an older application that has not needed to be changed for several years. Fortunately, you are an expert in the language it is written in. What are four things you would try to do on the first day you start to work on this project?

[2] (b) Your boss tells you *"don't make any changes until you understand the application"*. What is wrong with this idea?

[2] (c) What could you do instead that would probably make your boss happy?

2. XP

6 (a) Describe three potential benefits of pair programming..

4 (b) Describe two potential problems with pair programming.

4 (c) Describe the XP approach to making a schedule..

3. Tools

2 (a) What is the purpose of measuring test coverage?

3 (b) Explain how a *debugger* is used in debugging.

3 (c) Explain how a *debugger* is used in reverse engineering.

4 (d) Suppose that on your project, all new features were described by writing bug reports in a tool like Bugzilla. Suppose you wanted to make it easy to find all the test cases that were associated with a particular feature. Describe a process that would make it easy.

4. Architecture

4 (a) Two CS majors come back to Urbana for homecoming, 5 years after they graduated. They are talking about what they learned since graduation. One says he has learned how important it is for a project to have a good architect. The success or failure of projects he has been on are due to whether they had a good architect. The other is surprised, since where he works, most projects are successful and they almost never have an architect. Give two possible explanations for how they can see such different results.

4 (b) As a system gets larger, the way you develop the software changes because it takes a larger group of people to develop it. Describe two things about the way software is developed that change as the group-size grows.

5. Metrics

[3] (a) John is one of the fastest programmers in your group. However, Bob thinks John is careless and is creating more than his share of bugs. Describe a process for measuring the number of bugs created by each programmer on the project.

[2] (b) Suppose that you discovered that in the last month, John wrote twice as many bugs as Bob. Does that prove that John is a careless programmer? Why, or why not?

[3] (c) Describe the metric "response of an object". Why does a large number imply that a program is complex?

6. Documentation

[3] (a) Parnas and Clements list seven reasons why top down design of software is impossible. Give three of them.

[4] (b) They mean two things by "faking" top down design. What are they?

[6] (c) Prof Johnson gave the outline of system documentation. Parnas and Clements had a slightly different outline. Describe either outline, and explain what each section is.

3 (d) A user manual and a system design document have different audiences. What are their audiences, and how does this affect the way that documents are written?

2 (e) What is internal documentation, and in what way is it better than external documentation?

7. Design patterns

[4] (a) In the Visitor pattern, a Visitor object will visit each member of a collection. It is an alternative to adding a method to the class of each member of the collection. Visitor is not very common. Describe two characteristics of a system that would tell you that you should not use Visitor.

[2] (b) Suppose you were working on an application in which all GUI objects came from a particular framework. You have a spreadsheet object that you would like to use, but it is not part of the framework. What design pattern is most likely to help you reuse the spreadsheet object with the framework?

8. Putting it together

6 (a) Suppose you work at a company that builds tractors. They have a CAD (computer aided design) tool that models tractors, and helps design them so they are easy to manufacture. The tool is written in Java, and has classes for representing the various parts of a tractor. There is an abstract class `TractorPart` and subclasses for a transmission, a steering wheel, a windshield, and so on. A part like a transmission is made up of other parts, while a windshield is a single piece of plexiglass. So, each `TractorPart` could have an array of other `TractorPart`s as its "children". However, some designers would mistakenly give a part to something like a windshield that was not supposed to have any parts. So, the programmers added a `canHaveChildren` flag to `TractorPart`. The result was an addChild method like

```
1 public void addChild(TractorPart child) throws IllegalAddException {
2     if (!canHaveChildren) throw new IllegalAddException();
3     children.add(child);
4     child.setParent(this);
5 }
```

It is possible to eliminate the `canHaveChildren` variable by following the Composite pattern more carefully. Describe how you would refactor both TractorParts and its subclasses to follow this pattern.

10 (b) Before you refactor the program for the tractor company, you look to see which tests you should run, and there do not seem to be any tests of `addChild`. Design some tests for `addChild`. For each test, sketch pseudocode (or give JUnit if you want) and explain the rules/principles of test design that you are following.

6 (c) Management at the tractor company wonders why they are continuing to pay for maintaining a CAD tool when there are so many they can buy. In fact, most CAD tools have better user interfaces. They do not have all the specialized support that is in your tool, such as calculating the carbon footprint of manufacturing, or calculating how much of a product is imported, and from what countries. However, one of these other CAD tools has a scripting language that lets you write functions like this, and another outputs a design in XML so you can write programs to perform these extra features.

Describe the pros and cons of maintaining your own tool versus buying one. This is a classic problem in software reuse, so give the classic answers.