Information and Computer Science 52
Introduction to Software Engineering
Fall Quarter 2001

Name	__I. Am. Solution_______
Student ID	______________________________
Signature	______________________________
SEAT #	______________________________

Instructions

1. This exam is closed book, closed lecture notes, and closed personal notes. No PDAs, cell phones, SMS devices, …

2. This exam consists of 15 questions on 7 pages; answer all questions on all pages.

3. The questions are **not** equally weighted. The relative point value of each question is given at the beginning of the question. The total numbers of points that can be earned on this exam is 63 (oddly enough, the exact same as the midterm).

4. Write your name, student ID, seat number, and sign the exam where indicated above. In addition, write your name on **each** page of the exam. Failure to do so results in a loss of points.

5. Before you cheat, consider the ramifications: you will score an F for **all** of ICS 52, not just this exam.

6. Before you cheat, consider the chances: multiple eyes will be watching you from multiple locations in the room.

7. All answers are to be legible. Write clearly.

8. All answers are to be given on these pages. Use the backsides of the pages **only if necessary**, with a reference as to where any extra portion of an answer is located.

9. You have one hour and fifty-five minutes for this exam – that is, until 5:55p.m. Plan your time accordingly.

1. (10 points) List the 10 architectural idioms (styles) covered in class. You do NOT need to provide a description: just the name will do.

1 point per named style

These styles were given in the lecture notes and were on the midterm.

1. Batch sequential .
2. Pipe-and-filter
3. Data and/or service-centric systems: the Client-Server style, a.k.a. database centric design
4. Hierarchical systems/Main program and subroutines;
5. Data abstraction/OO systems
6. Peer-to-Peer
7. Layered systems
8. Interpreters
9. Implicit invocation (event-based)
10. Three-level architectures

2. (2 points) What's the purpose of the *integration* test plan?

To detect mismatches *between* modules. (E.g., to detect whether one module's provided interface matches up semantically with another module's required interface.)

WRONG Answers: to find bugs INSIDE a module.

3. (2 points) What's the role of **connectors** in a software architecture?

Any one of the following answers are correct, and is sufficient:
To serve as the locus for communication between modules.
To decouple modules from each other.
To provide a basis for examining communication between modules.
To support inter-module communication
…and so on…

4. (4 points)
(i) What's the point of identifying subset/superset systems in the requirements specification? 2 points for this part of the question (taken directly from the midterm)

Provides the basis for developing a solution architecture that will be capable of supporting incremental delivery and incremental development: the subsets can be designed and developed (and deployed) first, without requiring that the whole system be built in one single activity.

(ii) Suppose feature A belongs in the smallest subset system and feature B belongs in the largest superset of the system. Where should the modules in the architecture that implement feature A appear in the **uses** hierarchy as compared to the modules that implement feature B?

2 points for this part of the question.

The first essential point is that A's modules should appear at the bottom of the uses hierarchy (level 0, 1, …) and B's modules should ALL be above them. That is, there should not be a uses relation from an "A" module to a "B" module. But of course it is fine for B's modules to use A's. Second key point: Since B's modules belong to the largest superset they should be at the highest levels of the uses hierarchy.

5. (2 points) If you anticipate that a clean, straightforward implementation of your software will be too slow, when should you make the necessary optimizations to the code and on what basis should you make such optimizations?

   1 point for when, 1 point for basis
   The optimizations should be made only **AFTER** a clean implementation has been created and tested. In other words, no optimization before knowing that you have a complete, correct implementation.

   The basis for the optimization must be **measurement**: careful study to determine exactly where the performance problems lie.

6. (3 points) Explain the differences among unit testing, integration testing, and acceptance testing.

7. (5 points)
(i) Describe the Spiral model of software development. Use a diagram to illustrate your description.

(ii) Briefly describe three ways in which the Spiral process model is superior to the Waterfall model.

8. (2 points) Describe a development circumstance (a set of conditions) under which it would be better to follow a waterfall model of development, instead of the spiral model.

9. (8 points) Software engineering techniques and processes are not appropriate for all development projects.
(a) List at least four characteristics of a project that would argue FOR using the software engineering approach.

Grading: 1 point per characteristic, just like the midterm. Detailed answers are not required, just a listing of appropriate characteristics.

Example characteristics: Size of project (big), number of people involved (>1), external requirements imposed, family of products required, many changes over time requested or likely to be requested, costly, large/major consequences as a result of the development.

(b) For each of the characteristics you list in part (a), describe at least one way software engineering attempts to address the concerns raised by that characteristic.

Grading: one point per characteristic. The technique cited or approach listed here must be appropriate to the characteristic listed in part a. If you have any questions, just flag it and I'll check it.

Example answers: Size [applying software process to the development; breaking the problem into parts using modularity; …], Number of people [process], external requirements [requirements document, acceptance test plan, reviews], family of products [software architecture], changes [design by information hiding, software architectures], costly [process model, reviews, reuse], consequences [quality assurance techniques]

10. (3 points)
(a) Why *should* all the details of an application's *user interface* be included in a requirements specification?

1 point. Because such details are externally visible, the user most likely cares about them, and such details DO describe what is required of the software.

(b) What makes it seem *inappropriate* for including those details?

1 point. Such details emerge from a design process, namely the user interface design process. Thus design would have to go on during requirements.

(c) Which software process model best supports development of applications with successful user interfaces, and why?

1 point. Several do. The Spiral model and any incremental model do. They work because the process allows users to state their initial goals and requirements, all the developers to work ahead a bit (do an initial UI design, e.g.), and then give the users the opportunity to review and rethink what they want.

11. (2 points) Is it desirable to have a USES diagram with cycles? Why or why not?
It is NOT desirable, because that means that in order for any one of the modules involved in the cycle to work, they ALL have to work.

Another way of saying it: testing of one of those modules cannot proceed until all the modules involved in the cycle are ready for testing. And then they all have to be tested together.

12. (4 points) Name four coverage criteria used in structural testing. If you use the standard names (those in the lecture notes or in the book) you do not need to define the criteria. If you use non-standard names, then define the criteria.

1 point per criterion listed. If you get any odd answers, or ones you are not sure about, just flag it and I'll take a look.

a. node coverage. aka "statement coverage"
b. edge (or branch) coverage
c. loop coverage
d. path coverage

Questions 13 to 15 pertain to the following scenario; keep this description in mind as you answer the questions.

Your company, Crazy Software Ventures (CSV for short), is at it again. This time they have been awarded a contract to develop a replacement for the despised UCI TELE system. Specifically, CSV is tasked with taking TELE into the Web generation. This new generation of TELE will be accessed primarily by the WWW, though phone access via the current system will be preserved for future use, to accommodate those students who do not have web access at the time they want to register for classes. Interfaces to other legacy (existing "old") systems must be maintained as well, such as an interface to the tuition and fees processing system, the campus course rosters program, and the campus room assignment system.

13. (6 points) Designing the (student) user interface will be an important part of your activities. How will you go about that task? How will you assure yourself, the campus administration, and your student friends that you have done a good job?

14. (6 points)
(i) What do you think are the top three non-functional qualities your system must exhibit, and why did you pick these three? (Please ignore the user interface for this question.)

(ii) How will you go about assuring everyone that you have established these qualities in your implementation?

15. (4 points) What will you emphasize during the requirements analysis part of your process (once again, please ignore the user interface issues)?