

ICS 121 / Informatics 111 - Software Methods and Tools
Midterm Exam – Fall, 2005

Last Name: KEY First Name: _____

1. (20 points, 5 points each) Define the following terms, as used in software engineering:

Critical Path.

A sequence of activities without slack time.

Software Process.

The steps that lead to creating or enhancing the programs, procedures, and documentation associated with a computer system. (2 pts off for defining a software process *model*.)

Optimistic CVS Control Regime.

The CVS model where files are not locked on check-out, and conflicts are resolved on check-in. (3 pts off if no mention of resolution on check-in.)

Refactoring.

Improving the design of previously written code.

2. (15 points) According to *Using a Defined and Measured Personal Software Process* by Watts Humphrey, the "Personal Software Process" (PSP) is a "framework of techniques to help engineers improve their work." The four major steps are called PSP0 (which includes PSP0.1), PSP1 (includes PSP1.1), PSP2 (includes PSP2.1), and PSP3. Each step continues the elements of prior steps, plus new concepts and practices. Next to each of the terms and phrases below, write 0, 1, 2, or 3 to indicate the PSP step in which it is introduced or with which it is associated.

- | | |
|--|---------------------------------|
| <u> 0 </u> Process improvement proposal form | <u> 2 </u> Design reviews |
| <u> 1 </u> Test report | <u> 2 </u> Design templates |
| <u> 1 </u> Task planning | <u> 1 </u> Schedule planning |
| <u> 0 </u> Time and defect recording | <u> 1 </u> PROBE method |
| <u> 3 </u> Scaling up | <u> 0 </u> Size measurement |
| <u> 2 </u> Personal quality | <u> 3 </u> Cyclic development |
| <u> 1 </u> Size estimating | <u> 0 </u> Coding standard |
| <u> 2 </u> Code reviews | |

3. (12 points) In chapter 17 of *The Mythical Man-Month* (which you were not assigned to read), Brooks notes that some critics of the original "No Silver Bullet" article say it was focused on *productivity*, the software output per unit of input. One critic, Casper Jones, says, "No. Focus on *quality*, and productivity will follow."

- (6 points) Does it make sense to define "productivity" in this context as "the software output per unit of input"? What, specifically, are the "output" and "input" that Brooks has in mind to measure productivity?

Output: lines of code; test cases; documentation; training;

Input: person months

- (6 points) Do you agree with Casper Jones? Explain why or why not (argue one way or the other). Give a specific example, perhaps from the spectacular software system you wrote about, to bolster your argument.

3 pts: why or why not. If yes, for full credit, must explain "productivity will follow"

3 pts: specific example.

4. (8 points) In "The Mythical Man-Month," Brooks writes, "What does one do when an essential software project is behind schedule? Add manpower, naturally... This may or may not help." What quality of software tasks makes it often not helpful to add manpower?

For reference: "Men and months are interchangeable commodities only when a task can be partitioned among many workers *with no communication among them*" (Brooks, p. 16).

A full credit answer focussed on the software *tasks*.

5. (10 points) In *They Write the Right Stuff* by Charles Fishman, the main topic is how the shuttle software group is able to write large, almost defect-free software. According to the article, "How do they write the right stuff? The answer is, it's the process. ... The process can be reduced to four simple propositions." The first of these propositions is

1. The product is only as good as the plan for the product.

(5 points) Write down one of the other three propositions (here are some hints: subgroups and subcultures; master history; the process is so pervasive).

2. The best teamwork is a healthy rivalry.

3. The database is the software base.

4. Don't just fix the mistakes – fix whatever permitted the mistakes in the first place.

(5 points) Now describe one software method or tool that assists in following the proposition you named. (If you couldn't remember another proposition, you can use proposition 1.)

1. Write extremely detailed specs and design; "don't change the software without changing the blueprint."

2. Formal inspections of code.

3. Keep track of all code history, and all errors from the past.

4. Update the process when errors slip through.

6. (15 points) Many models for estimating the size of software look like this:

$$E = \alpha + \beta KLOC^\sigma$$

(5 pts for each part.)

- a. What does E mean in this equation?

Effort; or person-hours.

- b. Why might it make sense to raise $KLOC$ to a power greater than 1.0?

E increases faster than linearly with $KLOC$.

also ok: That's the value of sigma derived from the data.

- c. Name one big weakness of using $KLOC$ in an equation such as this.

**$KLOC$ is not known before the coding is done;
 $KLOC$ can vary by programming language.**

7. (20 points) Suppose that with your shiny new Bachelors degree and a few software courses under your belt, you have just been hired by the local startup company, Heartronics. Heartronics' first project is to write the software to control a new generation of pacemakers. (A pacemaker is "an electronic device that is surgically implanted into the patient's heart and hest to regulate heartbeat" according to the glossary at www.health.uab.edu/show.asp?durki=23563.) You report to the Vice President of Technology, who says to you, "Before we start development, I want you to design a process for our software engineering that combines the best of CMM and XP. Can you do that?" Respond (either "yes" or "no") with a brief memo that shows you really know CMM and XP and how they might or might not work together. Be specific and concise. Discuss at least one software tool or method.

General rubric:

- **5 points for responding yes or no with a well thought-out answer. Bonus for answering in the context of Heartronics (e.g., people's lives are depending on this software).**
- **5 points for demonstrating knowledge of CMM.**
- **5 points for demonstrating knowledge of XP.**
- **5 points for discussing a software tool or method.**

The rubric was not strictly adhered to, and an answer strong in one of these categories made up for weakness in another. If your page has four numbers totalled in the lower left, then those four numbers are points corresponding to the four categories of this rubric.