Name: Prof. Penny Student #:_____

CSC444F'11 Midterm Test

50 minutes – No Aids Allowed – 100 points total

Answer all questions in the spaces provided. Use the backs if you run out of space.
Write your name and student number on each sheet.

1. (20 marks) Name the "top 10" practices below (in any order) with a brief description of each, highlighting why it is important.

Source Code Control	"go-to" location for source code, enables collaboration and maintenance streams
Defect/Feature Tracking	Keeps track of work items, allows a process to be put around implementing them
Reproducible Builds	Ensures consistency in reproducing defects from the field, enabler for next item
Automated Regression Testing	Improves quality, allows for dangerous and last minute changes to be put in with greater confidence
Release Planning	Provides predictable quality, feature set, and end dates.
Feature Specification	Ensures work is done correctly the first time.
Architectural Control	Ensures the architecture remains good so that coders don't lose productivity in the future because of poor architecture.
Effort Tracking	Allows estimates to be improved over time
Process Control	Ensures a consistent process is used for all products
Business Planning	Ensures adequate funding is available to achieve business goals

2. (10) If a developer says they can be done coding and unit testing a feature in 5 ECDs during the coding phase, 90% of the time, is this sufficient information for a stochastic estimate? If not, what more would be needed?

No. With only one confidence interval cannot fit the estimate to any reasonable distribution, such as a Normal. Need at least two CI's, for example a 50% case and the 90% case given. In this way, we know the "spread" of the distribution (standard deviation) as well as the position of it.

Name: Prof. Penny Student #:_____

3. (10) Why is it reasonable to assume that F, the estimate of the total ECDs is, Normally distributed?

   The Normal distribution is unique in that the sum of a large enough number of stochastic variables drawn from any independent distributions whatsoever will tend towards being a Normal distribution. Since F is made up of the sum of many little f independent feature estimations, F is expected to be close to a Normal distribution.

4. (10) During class, a scheme was proposed to solve the problem of simultaneously having a good margin of safety on a release, yet still keeping developers productive. Describe this scheme.

   - Divide the features into two lists, A and B.
   - A are the important features, B are the nice to have features.
   - Set T so that the A list has a high probability of being done on time, but A+B has a low probability of being done on time.
   - Ensure developers work on A features before B if they at all can.
   - Reward the developers for getting as far as they can down the B list.

5. (20) Explain what can go wrong if one starts putting features into point releases, and why this happens.

   The point releases may start getting worse and worse in quality, possibly affecting many customers, whereas the new features are generally only of benefit to a small number of customers or prospects.

   This is because every time code is added, defects are introduced in some ratio. If for every 5 defects fixed, 2 new ones are introduces, the trend is still towards ever increasing quality.

   However, if code due to new features is added as well, the number of defects introduced may exceed the number fixed in the point release, and hence the quality will get worse and worse when customers are expecting it to get better and better.

6. (30 marks) Assume developers were taking a lot of sick days and this was causing problems in releasing software. Redefine attributions (mathematically) within the capacity constraint to better estimate and monitor this factor.

The capacity constraint is

(1) $F = N \times T$

(2) $N = \sum \frac{h_i}{8T}$

where h_i is a measure of the number of uninterrupted hour equivalents spent coding new features into the new release during the coding phase.

In the attribution given in the course,

(3) $N = \sum \frac{w_i t_i}{T}$

(4) $w_i = \frac{h_i}{8t_i}$

(5) $t_i = d_i - v_i$

Where d_i is the number of days that developer had available during the coding phase, and v_i is the number of workdays of vacation they took during that time.

To account for sick days, redefine the work factor, w_i, to add back in time taken as sick days by replacing (4) by (4') below. This gives a w_i unaffected by sick days, which is the first step in separating out the impact of sick days from other types of productivity losses.

(4') $w_i = \frac{h_i + sh_i}{8t_i}$

where sh_i is a measure of the number of hours of sick leave they took when they were officially supposed to be at work.

If we leave it at this, we can no longer solve (3), (4'), and (5) to get back to (2), which is still the most desirable definition for N, measuring total work done towards features.

Therefore introduce a new term s_i analogous to w_i, corresponding to the daily propensity towards sickness, into (3)

(3') $N = \sum \frac{w_i t_i - s_i t_i}{T}$

Equating (2) and (3'), we can solve for the definition of s_i to be

(6) $s_i = \frac{sh_i}{8t_i}$

With both w_i and s_i now available, we can independently predict and monitor the impact of sickness on the developers as opposed to other types of productivity losses. For example, a developer may have a new w of 0.7, and an s of 0.1 (which means on average they are sick for 0.8 hours per official workday), which is equivalent to an old w of 0.6.