UNIVERSITY OF TORONTO
FACULTY OF APPLIED SCIENCE AND ENGINEERING
FINAL EXAMINATION, December, 2009
CSC444H1 F – Software Engineering I
Examiner – D. Penny
2½ Hours - 100 marks total – each question worth 10 marks
Write all answers in the exam books provided.
No aids allowed other than non-programmable calculators.

Answer all questions succinctly, but including all relevant points. Think deeply about your answers, demonstrating your insight and understanding of software engineering with each one, and make convincing arguments to back them up. If you make a statement, for example "a certain practice is inefficient," then be sure to explain why. Avoid parroting back stock phrases from the book. Good luck!

1. Suppose a coder with a work factor of 0.6 estimates "2 weeks" for a feature, but implicitly is thinking of elapsed calendar time. How many ECDs will the feature take?
    - 2 calendar weeks generally equals 10 workdays (4) (but not always (1)).
    - 10 workdays * 0.6 ECDs/workday = 6 ECDs (5)
    - If they say 10 ECDs, it may be right, but they must explain that the person was neglecting to consider their own work factor when coming up with the estimate.

2. Suppose a release is planned such that D(50) = N(400,100) (a Normal distribution with mean 400 and sdev 100. Recall that the probability encompassed by +/- 1 sdev is about 68%). Make statements about the likelihood of the release coming in on time, 100 ECDs ahead of schedule, and 100 ECDs behind schedule.
    - Note that to be precise, releases do not come in a certain numbers of ECDs ahead of or behind schedule. At dcut, we may have a certain number of ECDs left to do, or we might get to dcut earlier by 100 ECDs / sum(work factors) (1)
    - There is a 50% chance the release will come in on or ahead of schedule. (3)
    - Note: for the above they may say there is "almost no" chance the release will come in exactly on time - give that 3 as well
    - There is only a 16% the release will come in earlier than 100 ECDs ahead of schedule (3)
    - There is an 84% chance the release will come in no later than 100 ECDs behind schedule. (3)

3. Explain the problems with a Normal distribution as an estimate for *w*, the work factor. Why might we choose to use one despite these problems?
    - A Normal has non-0 probability of being negative or much larger than is sensible, w does not (2)

- A Normal is a symmetrical distribution. The distribution for w might be skewed. (2)
- We might choose to use a Normal because it is easy to compute with (2)
- We might choose a Normal because you only need 2 numbers to define one. (2)
- We might use it because the error introduced by using a distribution that approximates w is much less than other errors in release planning. (2)

4. While the purpose of release planning is estimating things in advance, why did we spend so much time defining how to measure what *w* and *f* were after the fact?
    - If we define how to measure these quantities, than we have defined them completely. This provides a solid definitional basis for what we are estimating. (5)
    - By defining how to measure them, we can measure them and compare the actual measurements to the estimates in advance, thereby allowing us to improve our estimation accuracy over time. (5)

5. Describe what they are, and explain how and why the main, maintenance, and shipping codelines are used.
    - The main codeline is used for all new feature development, but only from fork to dcut. (1)
    - All defect corrections may be checked into the main codeline at any time (1)
    - A maintenance codeline is forked from the main codeline (1)
    - It is forked when development is ready to begin on the next feature release. (1)
    - A maintenance codeline is used to stabilize a release prior to shipping it. (1)
    - A maintenance codeline is used to enable bug fix maintenance releases while new feature development continues on the main codeline (1)
    - Only defect corrections may go into a maintenance codeline (1)
    - A shipping codeline may be branched from a maintenance codeline immediately prior to making a shipment. (1)
    - It is used to allow general defect correction to continue on the maintenance codeline (1)
    - It allows us to enforce that only specially selected last minute defect corrections are allowed into the shipping codeline. (1)
    - Other points are possible - give +1 for a reasonable one.

6. Continuous integration means building the product every night and running a set of sanity tests against it to check for problems during the coding phase. Why is continuous integration considered to be a best practice?
    - Continuous integration helps finds defects closer to their point of injection (2). If one can catch a defect closer to its time of injection, the cost of fixing it is lower (3)

- Continuous integration allows developers to work with the latest changes from all other developers (1), without creating an unstable environment for themselves (1). Working with the latest changes avoids a situation where a coder feels a feature is working, but later on, after it is integrated with others work, it is found not to work (1). This can lead to unexpected schedule delays later in the release cycle. (2)

7. In class it was stated that "testing is the easiest group to manage". Explain the reasoning behind this statement.
    - There is a relatively easy metric than can be applied to the testing group: the coverage of all the tests that they do. (5)
    - One can manage the group by checking that the coverage metric is continuously going up. (5) - or something close to that

8. Describe the key elements of the Scrum Agile programming methodology.
    - Sprints are 2-3 weeks cycles where the software is ready to be shipped at the end of each sprint. (2)
    - All potential features are e coded in 3x5 cards as "stories" (1)
    - The coding group sizes each "story" (= feature) in "units". (1)
    - Coders commit to being able to code a certain number of "units" on each sprint. (1)
    - Coding teams will adjust their predictions of number of "units" they can code on a sprint after each sprint. (1)
    - The stakeholders get to choose what features to work on for each sprint, subject to the sum of the units being less than what the coding team can handle during a sprint. (2)
    - The stakeholders gets to use the software after every sprint. (1)
    - Daily stand-up meetings are held (called "scrums") to assist the coders in removing obstacles. (1)
    - Other points are possible as well - give 1 for a good solid one (not just some general goodness principle).

9. Why is a workflow system for features important for managers?
    - It allows us to keep track of all the requested features, and choose the ones for inclusion in the next release. (2)
    - It allows the managers to prioritize which features are to be worked on at what point during the release cycle and by whom. (3)
    - It allows managers to measure the progress of features through the defined software development process, by reporting on e.g., how many features in a release are in what state. (3)
    - This enables the managers to see the extent to which their defined processes are being followed. (2)
    - There may be other good reasons - give +1 for other good plausible ones

10. What is a "non-functional requirement?"
    - Golnaz: over to you for an answer!