FINAL TEST (100 pts)
CPSC 544
Spring 2020
DUE: Before 11:59 pm of Sunday, Dec 13, 2020

Dr. Bin Cong

NOTE:

1. **Email me at bcong@fullerton.edu if you have any questions between 9:30 – 11:30. After that, I may not be able to answer questions in real time.**
2. **Make necessary and reasonable assumptions while developing your solutions if needed.**
3. **Make your answers clear, precise, and to the point.**
4. **List references if used.**
5. ***DO NOT DISCUSS SOLUTIONS WITH ANY OF YOUR CLASSMATES. VIOLATION OF THIS OR ANY FORM OF CHEATING WILL AUTOMATICALLY RESULT IN AN "F" FOR THE CLASS.***
6. **Submit the solutions before 11:59 pm on Sunday, Dec 13 by emailing me the solutions at bcong@fullerton.edu and also submit it on Titanium. Please include your first and last name in your solution filename.**

STUDENT NAME: Yifan Wu
ID # : 886654896
SECTION NUMBER: 01

Question 1 (12 points): Software Process Maturity Related Questions （Justify your answers）

1) Humphrey pointed out that "Software process changes won't stick by themselves". Which maturity level ensures the changes brought by improvement will stick?

   Level 5 Optimizing Process

   From chapter 1, the level 5 ensures the changes will stick, because with Level1-Level4, the organizations has achieved predictable schedules, a stable process with a repeatable level of statistical control, defined the process as a basis for consistent implementation, and initiated comprehensive process measurements and analysis, which are when the most significant quality improvements begin.

   So after level 4, in level 5, the organization now has a foundation for continuing improvement and optimization of the process. It has enough preparation to ensure the new improvement changes will stick and run well.

2) Why software defect prevention is placed at the level 5 (optimizing process) given it seems to be a simple work.

   Since software defect prevention seems to be a simple work, even though it's always possible to do more reviews, to run more tests, or put it to other levels, however, it will cost both time and money to do so, what's more, with the complexity of the systems increasing, it will make the systems progressively more difficult to test. Only with level 5 that the data is available and the information is prepared, to understand the costs and benefits of such work. The level 5 provides the foundation for significant advances in software quality and simultaneous improvements in productivity. It will save time and money to do defect prevention on this level, before it goes too complex and too late to save.

3) Can an organization achieve CMMI Maturity Level 5 mainly using Agile? Why?

   It can't mainly use Agile to achieve Level 5. It should mainly use CMMI to achieve this process.

   CMMI is a process improvement approach, to be used to access organizational processes, set improvement goals and provide guidance for process improvement. The main goal of CMMI is organizational improvement for existing processes.

   Agile is a software development methodology that breaks down the development process into interactions(sprints). The goal is to produce a shippable product that can be handed over to a customer. It's focused on the product quality, but not included risk, quality and other management processes.

In contrast to CMMI, Agile is not focused on the existing processes, but instead on creating new processes and products. By comparing these two terms, we know even though agile might be able to optimize the project with future deliveries and achieve part of level 5, but it's not used for existing processes, it can't mainly use Agile to achieve level 5.

4) Name three practices which should be included in the maturity framework discussed in the textbook given what we know now.

1. Improvements should be made in small steps.
   People do not naturally perform well on highly rigorous and complex tasks. Software engineers must be tailored to their abilities, rather than the reserve.

2. The organization should train the employees.
   Without training, the necessary learning is gained through trial and errors. This not only wastes time and money; it also often involves a lot of errors and it will damage the software process.

3. Someone must start to work on it to improve the software process.
   Some one must consider raw materials handling, design the process flow, select the tools , specify the controls and overseas ordering, installation and operation.

Question 2 (18 points): Software Inspection

Develop a code inspection (review) process based on what is given in the book, or what you read in some other places.
What metric you will use to assess the inspection performance and for planning the future review.

Process:

1. Planning.
   First step, verify if the materials meet entry criteria.
2. Introductory meeting
   Second step, the code author presents the materials, and explains goals and rules.
   Schedule next inspection meeting.
3. Inspection meeting
   Third step, the inspection group reviews the materials, and logs the defect.
   The inspection recorder collects report matrices.
   If no defects are found, then complete the inspection; else go to the next step.
4. Rework
   Fourth step, the code author debugs and fixes defects, then collects metrics.
   Schedule the next verification meeting.
5. Verification meeting
   Fifth step, the reviewer verifies defects if it's fixed.
   If additional defects are found, go back and repeat the fourth step.
6. Complete
   Can have a follow-up meeting if needed to discuss how to improve the inspection process in the future for the organization.

I will use TABLE 10.7 to assess the inspection performance and planning future reviews.

Release	Relative hours/KCSI			Relative hours/Defect			Inspection Efficiency	Quality Index
	I0	I1	I2	I0	I1	I2		
Release1								
Release2								
Release3								

*
KCSI = thousands of changed source instructions
I0 = High-level design inspection
I1 = detailed design inspection
I2 = code inspection
Inspection Efficiency = (defects found by inspections) / (total defects found by inspection and test)
Quality index = (defects found in field use) / (total defects found during development)

The result of "Inspection Efficiency" can help my future planning.

Question 3 (10 points): Software Quality Measure

Why is “Survey” (to measure Usage) not a “good” measure? Why “Defect” is such a popular measure?

To measure usage, “survey” is not a good measure. Surveys are difficult to be available and it’s not controllable.

Survey needs to cost money to invite people to join the survey, and it’s not efficient since most people are not willing to do so for free or little rewards. Then, even if people are willing to do so, the survey data is not accurate enough because people will have bias to measure software usage; in contrast they need to spend time to use the software in order to estimate the usage.

While defect is a popular measure. Compared to surveys, defects are highly available and controllable. I use TABLE 16.2 as a reference, instead of using survey, defects can be a better way to measure usage. Statistically, usage actually had a slight negative correlation with program defect rate; that means, within a certain point, more defects will cause less usage, meanwhile less defects can measure more usage.

4 (15 Points): SQA and the Axioms of Testing

1. What did Humphrey suggest to get the right people performing SQA duty? Do you agree with him in today's environment? Why? (5 points)

   Humphrey suggests that one effective solution to get the right people performing SQA duty is to require that all new development managers be promoted from SQA, which means potencial development managers need to spend at least 6 months in SQA, before being development managers. By this solution, SQA work will have full management backing and it will be effective.

   I agree with Humphrey because I have experience that the key points to make SQA be effective is to have the full backing of management support. Even in today's environment, some managers still prefer earlier delivery, instead of less defects.

   When I was in a startup company, we didn't have enough SQA since the budget. As the deadline approached, there were too many pending tasks not done yet, and the manager asked SQA to do development jobs, to make the project deliver sooner. Then, the manager asked the marketing department to test and do the SQA jobs. However, after releasing, high defects rate causes low software usage, and the company can't make profit by few customers. By this example, the manager was not fully backing the SQA and causing trouble.

2. Justify 5 testing axioms given in the textbook (Page 194 section 11.2.1

   1. The good test case is one that has a high probability of detecting a previously undiscovered defect, not one that shows that the program works correctly.

      When developers write a program, the developers will test the outcome they expected. If they find a different outcome or the outcome is not expected, or the program crashes, then they will fix and debug the program. These kinds of bugs are discovered defects and developers might have found them already. So, a good test case is to find the undiscovered defect, which the developers don't think or recognize before. A good test case can save the program from unexpected faults.

   2. One of the most difficult programs in testing is knowing when to stop.

      Obvious test cases are easy to create, but edge test cases and stress test cases are hard to think of; even more, you don't even know if you get enough edge cases, because there might be more cases that are unusual and hidden, that's why it's difficult to know when to stop,

3. It's impossible to test your own program.

   When you write a program you always miss something because of human nature. Firstly, when you deliver the program you have tested many cases relative to your coding, because you want to know if your code is right. However, testing is not only focused on the code, but also focuses on putting the program in real life. Secondly, developers think so highly of their code and subconsciously they will think testing is unnecessary, so the testing will be inefficient.

4. write test cases for invalid as well as valid input conditions.

   Valid test cases can ensure the program works well in the right way, while invalid test cases can protect the program and secure it. Some invalid input might cause segment fault, memory overflow, or data leaking. For example, valid SQL test cases can ensure the output is correct, but invalid SQL test cases can prevent SQL injection hacking.

5. Assign your best programmers to testing.
   The better programmers, the more experiences they have, and they know what usual flaws might happen and they will think of different ways to break the program, so they can be better testers. They have bigger views of software architecture and they can think of more points to take care.

Question 5 (20 points) The Software Crisis is Dead!

The software crisis is dead! This is the 1st sentence of the textbook (Foreword by Peter Freeman). After the semester's study, do you agree with the statement? Please list 10 specific "best practices" in the software process that might help us resolving software crisis.

No. Software crises can be caused by human nature, and mistakes by humans can't be avoided totally. Even with the best management, the software crisis can be resolved as much as possible but can't be removed forever.

Best practices:

1. Use inspection report (TALBE 10.4, 10.5) to better inspect code.
2. Spend time carefully planning the project, even if it looks like wasting time.
3. Require all new development managers be promoted from SQA, to get better SQA.
4. Before establishing an aggressive standards development program, it's wise to formulate an overall plan that considers the available standard, the priority needs of the organization, the status of the projects, the available staff skills, and the means for standard enforcement.
5. Let SQA people only do SQA jobs.
6. Major changes to the software process must start at the top.
7. Manage, audit, and review the work to ensure it's done as planned. Reviewing the work isn't wasting time; it helps the software process.
8. Divide the work into independent parts. This can help software process by better managing a large project.
9. When the gap between your knowledge and the task is severe, fix it before proceeding, or it will cause a software process trouble.
10. Precisely define the requirements for each part before actually doing it, this will save time and avoid rework.

Question 6 (25 points): Short Answers

1. Use an example to illustrate the following statement: the quality of a software product largely depends on the process used to develop such a product

   For example, a good senior programmer develops a large project without good processes. this programmer finishes 50% of code, and then, the programmer leaves the organization. Since there are no good processes, the new hired engineer can't continue to work on it with good quality. So the quality of this software will be affected eventually, but with good processes the software can remain the same level of good quality from the beginning.

2. Give examples to illustrate the difference between Errors and bugs (Page 312 of textbook).

   When I click a calculator button to calculate 1 plus 1, the result is not 2, that's a bug, but if the program crashes, it's an error.

3. Name a tool that can be used to automate CM (Configuration Management) process, list the main features of the tool of your choice.

Puppet Configuration Tool

This scm tool Allows you to take full control and visibility over your software delivery process.

Allows you to make quick changes or remediate urgent problems alongside your model-driven automation management.

This configuration management tool helps you to manage infrastructure as code using your favorite version control systems

4. Name three challenges a SEPG will face.

It doesn't have enough full-time capable professionals to do competent work.
The SEPG doesn't have sufficient management support to convince the projects to support the process improvement efforts.
The SEPG manager is not able to obtain the participation of the most knowledgeable software professionals in the process improvement task group.

5. Name three reasons why accurate software estimation is hard to achieve.

Requiremental uncertainty. New requirement always shows up in the future so it's not included now.

Technical uncertainty. Some bugs might delay the project schedule since it's hard to find and spend much more time fixing it.

Staffing uncertainty. People are changing jobs and companies often in the software industry due to higher pay. If someone leaves the project then it will affect the schedule and it might affect the deadline and quality.