Donovan Moini
Dr. B.J. Johnson
CMSI 401
10/30/2019

Assignment 2

Problem 1

Write a short paragraph to answer these three questions:

- What are the two major concerns of any software project?
- Which do you feel is more important?
- Where does the idea of complete functionality fit with these two concerns?
    - The two major concerns of any software projects are its costs and the required time for completion. I believe the cost is more important of the two. If you need more time to complete a project, you can likely request an extension for more time. However, you cannot always do the same for cost. If your application relies on funding and depletes such funding, you cannot create more funding out of thin air. This lack of funding could lead to the demise of the application. The idea of complete functionality relates to the unrealistic idea of an application's cost and time are within the range of the budget and timeline respectively, in addition to expecting everything to work correctly.

Problem 2

- In the Agile method for software development, what are the four main phases that occur in each and every iteration? Do you feel that any of them could be done at the start of the project and not be repeated in every iteration? Do you feel that would save time overall on the project? Justify your answers with a brief explanation.
    - The four main phases are requirements, design, code, and testing. I believe that the requirement phase could be done at the start and does not need to be repeated at every iteration. The requirement lays out the roadmap of the project, and updating this at every iteration would constantly change the direction of the project. I do believe though it can be revisited at times to adjust the roadmap. I believe this would save time as the developers working on the project would have a sturdy understanding of the project. If the project's roadmap changed every week, the developers would have to adjust to a new roadmap every week, which would add time to the project.

Problem 3

- In the Waterfall method for software development, what are the main phases that occur? How are they different from the phases in the Agile method? What other phases are in Waterfall that are left out of Agile? Do you think these are needed in Waterfall? Describe a situation using Agile in which one of these extra Waterfall phases might me needed.
    - The main phases of the Waterfall method are requirements analysis, design, code, test, and maintenance. The Waterfall method includes the maintenance phase while the Agile method does not. The other phase left out of the Agile method

includes the deployment phase. I think the maintenance phase is needed because it ensures that the application will constantly be updated. Once an application is launched to the public, it always has some bug that is unknown to the developers. Over time, this bug and others become apparent from numerous users using the application. When the maintenance developers are notified of the bug(s), they can work on a hotfix or patch to fix the bug(s) and release an update to all users using the application.

Problem 4

- Write one-sentence answers to the following questions:
- What is a "user story"?
    - A user story is a tool used to gain information about the software from the customer's perspective of interacting with the software.
- What is "blueskying"?
    - Blueskying is a large scale brainstorming that captures the ideas of everyone that can add ideas to the software. All ideas are welcome as long as they focus on the core needs of the software.
- What are four things that user stories SHOULD do?
    - "Describe one thing that the customer needs the software to do for the customer" (39)
    - "Be written in language that the customer understands" (39)
    - "Be written by the customer" (39)
    - "Be short. Aim for no more than three sentences" (39)
- What are three things that user stories SHOULD NOT do?
    - "Be a long essay" (39)
    - "Use technical terms that are unfamiliar to the customers" (39)
    - "Mention specific technologies" (39)

Problem 5

What is your opinion on the following statements, and why do you feel that way:

- All assumptions are bad, and no assumption is a good assumption.
    - I do not agree with this statement. For starters, this is an absolute statement, which is usually incorrect. I believe that relying on assumptions is not good practice as it is risky, but there are times where assumptions can help an application. Assumptions can help think of edge cases and all types of users who will use the application. Even with assumptions having the potential to help an application, if you can check in with a user instead of relying on an assumption, go with the former.
- A big user story estimate is a bad user story estimate.
    - I agree with this statement. If you are thinking too large of a scale, this can lead to the issue of adapting to changes along the way of completing the application. If a user wants to change something, then the entire estimate will have to be updated rather than a specific component of it. The more times a user wants a change, the

entire plan will have to constantly be redrawn, and possibly lead to the incompletion of the application.

Problem 6

Fill in the blanks in the statements below, using the following things [you can use each thing for more than one statement]: Blueskying; Role playing; Observation; User story; Estimate; Planning poker.

- You can dress me up as a use case for a formal occasion: User Story
- The more of me there are, the clearer things become: User Story
- I help you capture EVERYTHING: Blueskying, Observation
- I help you get more from the customer: Blueskying, Role Playing, Observation
- In court, I'd be admissible as firsthand evidence: Observation
- Some people say I'm arrogant, but really I'm just about confidence: Estimate
- Everyone's involved when it comes to me: Blueskying
- NOTE: when you have finished, check your answers with the result in your text on page 62. Do you agree with the book answers? If you disagree with any of them, justify your preferred answer.
    - For" I help you get more from the customer," I argue that blueskying fits with the statement because if you can ask for ideas from the customer(s).

Problem 7

- Explain what is meant by a better than best-case estimate.
    - A better than best-case estimate is an unrealistic estimate that has a greater strain than the best-case estimate and has many underlying assumptions with it. For example, the book provides an example of asking a programmer to write a PHP interface for a SQL database. A programmer might say, "Sure, no problem, I can crank through that in two days" while in reality, the programmer will be thinking "I'll grab a Monster on the way home, program till 3 A.M., take a Halo break… and finish at midnight. As long as nothing goes wrong…". There are many assumptions that tie into this estimate, and these are only the ones the programmer is aware about. This expectation is incredibly unrealistic and more demanding than the best-case estimate.

Problem 8

- In your opinion, when would be the best time to tell your customer that you will NOT be able to meet her delivery schedule? Why do you feel that is the best time? Do you think that would be a difficult conversation?
    - I think the best time to tell your customer that you will not be able to meet the delivery schedule would be as soon as it is known. Firstly, sharing this information sooner is much better than doing so later. If a customer received this information on the delivery date, just imagine the frustration the customer would feel. Secondly, this will create a more transparent, communicative relationship between the customer because you will be keeping the customer in a constant loop of the application's development. I believe would be a difficult conversation

because the customer does expect for the delivery schedule to be met, but it is better to be honest about it sooner than later.

Problem 9

- Discuss why you think branching in your software configuration is bad or good. Describe a scenario to support your opinion.
    - I believe branching is an important aspect of software development because it supports multiple people working on the same stable code simultaneously. For example, in Amplify we have branches for each member to work on individual features. We have our master branch designated as the main branch that must always be working correctly. If I want to implement a new feature, I will create a branch from master and begin working on the feature on the new branch. If I want to save a half-completed version of this new feature, I can save it on my branch, which would not affect the master branch. Additionally, this keeps the new feature isolated from all the other features being implemented as well. If five different features were being worked on at once on the same code, it can be very confusing about what is being implemented. Additionally in this scenario, if I want to push my completed feature, I would either possibly push other incomplete features or have to wait until all other features are completed. Another example of why branching is good is we do not want to accidentally push broken code to our master branch. If somebody pushes broken code without realizing it to our master branch, that would break one of Amplify's features, or even Amplify itself. If the broken code was instead pushed within a branch, it could be caught and fixed before being merged into master.

Problem 10

- Have you used a build tool in your development? Which tool have you used? What are its good points? What are it's bad points?
    - For my senior project, Amplify, my group is using GitHub Actions. GitHub Actions allows for continuous integration and automated testing after pushing code to Amplify's repository. GitHub Actions is good because it ensures that all code pushed to any of Amplify's branches passes all the tests, and if not we are notified and can fix the issue. The bad point includes that we need good tests to ensure our builds are constantly stable. Currently, we have mock tests to just make sure GitHub Actions testing works, but we have to spend time adding numerous testing.