CMSI 401 Assignment 2

1. The two major concerns with any software project are time and cost. Both of these concerns limit the speed and quality of a software project. In my opinion, I believe time is the most important of the two. I believe this because a team could have all the money in the world and still not be able to finish a project due to the time constraint. However, if money was an issue but time was not the group could find other work arounds and complete the project regardless. This brings up the idea of complete functionality as you cannot have one without the other and a proper balance of the two is needed for a good project.
2. In the Agile software development method, the four phases that occur every iteration are Define, Build, Test, and Release. The only thing that could be done at the start of the project would be if you were to define everything in your software from the get-go. This could save some time on the project, but I don’t think it is very possible or realistic to know exactly how everything will work before building. I think there are lots of unexpected obstacles and the agile approach is better suited for development because it is more adaptable.

1. The Waterfall approach is very similar to the Agile approach in software development. Its main phases are requirements, design, implementation, verification, and maintenance. I believe the last four of these phases are synonyms with the Agile development process, so the only thing that waterfall requires specifically is the requirements. During the waterfall design process, the requirements are set before the work begins. This would also be needed in agile because every single iteration meets a requirement instead of listening them all out like in waterfall.
2. 1. User Story is a story about how the user interacts with the software.
   2. Blueskying is the idea of coming up with wild and crazy goals while still focusing on the core needs that the software is trying to meet.
   3. User Stories should describe one thing that the software needs to do for the customer, should be written using language that the customer understands, should be written by the customer, and should be short.
   4. User Stories should not be a long essay, should not use technical terms that are unfamiliar to the customer, should not mention specific technologies.
3. The two quotes “All assumptions are bad, and no assumption is a good assumption” and “A big user story estimate is a bad user story estimate”, are two things to keep in mind when developing software. The first statement is to ensure that developers consider all risks that their project might involve. The second quote talks about how estimates that are usually longer than 15 days provide a higher room for error. Both rules of thumb are used to make sure that both the customer and developer are pleased with the final project.
4. 1. You can dress me up as a use case for a formal occasion: **User Story**
   2. The more of me there are, the clearer things become: **User Story**
   3. I help you capture EVERYTHING: **Blueskying**
   4. I help you get more from the customer: **Observation**
   5. In court, I'd be admissible as firsthand evidence: **Observation/User Story** 
      1. I chose user story because I believe the idea behind the project might also come up in court as well.
   6. Some people say I'm arrogant, but really, I'm just about confidence: **Estimate**
   7. Everyone's involved when it comes to me: **Blueskying/Observation**
      1. I chose observation because while the customer might not always observe the project progress it is still possible.
5. The “better than best case” estimate just means that programmers think in utopian days. They believe that they can complete the project in a reasonably fast amount of time and predict nothing going wrong. In other words, this is a too good to be true estimate.
6. I believe the best time to tell a customer that you cannot meet their delivery schedule comes when the team has exhausted all options and it is 100% certain that they will not meet the deadline. If this point does occur, then it would be best to tell the customer right away and have a plan to complete the project within one additional iteration. This is the best time because there are no excuses and no other options to proceed with. I do believe it would be a difficult conversation, but I believe having a plan of attack will help ease the customer.
7. Branching is good when you have released a version of the software that you need to maintain outside of the main development cycle, you want to try some radical changes to code that you might need to throw away, and you don’t want to impact the rest of the team while you work on it. Branching is bad when you can accomplish your goal by splitting code into different files or libraries that can be built as appropriate on different platforms or when you have a bunch of developers that can’t keep their code compiling in the trunk so you try to give them their own sandbox to work in. Therefore, I believe branching is not worth the risk unless absolutely necessary as it just adds more problems than are needed.
8. We have used a build tool in our development called Visual Studios. It holds all our files and allows us to compile edit and update our code. I would say there aren’t any defects to the build tool as we can run our app from its terminal.