Senior Project Assignment 2

- Problem 1
  - **Write a short paragraph to answer these three questions:**
    - **What are the two major concerns of any software project?**
    - **Which of those two do you feel is more important?**
    - **Where does the idea of complete functionality fit with these two concerns?**

The two major concerns for any software project are "How much does it cost?" and "How long will it take?" We believe that either can be more important depending on the project, company, and state of the industry. For example, a startup that had a project that would've been super desirable for people who had to stay home during COVID might prioritize speed over cost in order to get their product out in time for it to be relevant. In contrast, a well-established company may take their time and prioritize cost for an update to an existing service. Through iteration, hopefully the customer and development team will discuss and agree on what complete functionality means with considerations for cost and time. The scope of the project may expand or contract dramatically based on what happens with funding, timing, or a combination.

- Problem 2
  - **Write a short paragraph to answer these three questions and briefly explain your opinion:**
    - **In the Agile method for software development, what are the five 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?**

The five main phases that occur in each and every iteration of the Agile method are: brainstorming/planning, designing, developing, testing, and deploying. None of these phases could be done at the start of the project; they must be iterated throughout the whole process. Requirements may change or unexpected events may set us back, forcing us to stray from a set plan. If we were to plan and design the project at the start without repeated iteration, it would be faster. However, this speed comes at the cost of lower quality.

- Problem 3
  - **Write a short paragraph to answer these four questions and briefly explain your opinion:**
    - **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 be needed.**

The main phases that occur in the Waterfall method are: requirement analysis, design, code, test, and maintenance. These phases are different from the Agile method in terms of flexibility. The Waterfall method is linear, whereas the Agile method is iterative. The phase that is left out in Agile that is in Waterfall is requirement analysis. This is because the Agile method plans ahead of requirement changes and constantly adapts through iterations. However, this phase is needed in Waterfall because the model is linear. We must know what is needed before designing the model. In situations where requirements are fixed, this Waterfall method may be useful.

- Problem 4
  - Write one-sentence answers to the following questions:
    - **What is a user story?**
      A user story is a story about how users interact with the software you're building.
    - **What is blueskying?**
      Blueskying is a brainstorming process where no idea is too large or preposterous - the sky is the limit.
    - **What are four things that user stories SHOULD do?**
      User stories should do the following:
      1. describe one thing that the software needs to do for the customer.
      2. Be written using language that the customer understands.
      3. Be written by the customer.
      4. Be no longer than three sentences.
    - **What are three things that user stories SHOULD NOT do?**
      User stories should NOT do the following:
      1. Be a long essay (3+ sentences)
      2. Use technical terms that the customers don't understand
      3. Mention specific technologies

        - **Does the Waterfall method have user stories?**
          No, the Waterfall method does not have user stories.

● 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 don't think that ALL assumptions are bad, especially in software development. It's impossible to have perfect knowledge, so we sometimes have to make assumptions to move forward. What makes an assumption bad is when it's left unspoken or never clarified with the customer. As the book explains, teams should "eliminate as many assumptions as possible by clarifying those assumptions with the customer." Any assumptions that remain should be considered a risk, and managed properly.

    - **A big user story estimate is a bad user story estimate.**

      A big user story estimate isn't necessarily a bad estimate. The book explains that if a story's estimate breaks the 15-day rule, the team has two options: either break the story into smaller, easier to estimate pieces, or revisit the story with the customer. Talking to the customer may clarify things, and those assumptions might go away, and even cut down estimates significantly.

● 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
    - I help you get more from the customer: Role playing
    - 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's answers? If you disagree with any of them, justify your preferred answer.

- Problem 7
    - **Explain what is meant by a better than best-case estimate.**
        - A better than best-case estimate is a project completion estimate that does not allow for any mistakes. A lot of assumptions are made without acknowledging potential problems they may run into.
- 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? If so, how could you make it less difficult?**

      In our opinion, the best time to tell the customer that you will not be able to meet their delivery schedule is as soon as you know that you will not be able to meet it. We feel that this is the best time because it affords the most time for re-evaluation and adaptation. We do believe that it would be a difficult conversation, but much less difficult than hitting the deadline and revealing that you've known for the past month or whatever that you weren't going to make it. It's entirely possible that the customer is willing to compromise on certain features in order to have a working product. In order to make it less difficult, we think it would be beneficial to have detailed information about why the deadline can't be met, possible paths the team can take to satisfy the customer as much as possible, and a good old fashioned apology.

- Problem 9
    - **Write a short paragraph to discuss why you think branching in your software configuration is bad or good, then describe a scenario to support your opinion.**

      Like many things, we think that branching is not inherently good or bad. Rather, we feel that branching is a useful tool that can be used intelligently to organize code and workflows or unintelligently to overcomplicate repos. With good version control hygiene, branching can be a great way to separate rapid development from stable releases. New features, big refactors, and R/D can be done in a relatively safe way on a branch. However, there is definitely the

potential for diverging branches that require vastly different testing, upkeep, and development. As long as the team understands the implications of starting a new branch and doesn't do it too often, branching can be extremely helpful.

- Problem 10
  - **Have you used a build tool in your development? If you have, which tool have you used? What are its good points and bad points — in other words, what do you like about it and/or dislike about it?**

    No, we have not used a build tool in development. In our development, we haven't gone any further than user guides and batch files to simplify installation and usage of software projects. Though we haven't personally used these tools, we can see the potential benefit they offer. Frontloading the setup work as a technical person can save tons and tons of time in the long run, especially when non-technical people may have to deal with the consequences. More than that, developers who haven't seen or touched your code would benefit from build tools since they wouldn't have to familiarize themselves with your codebase if issues arise.