Aidan Esposito, Matthew Savitt, Parker Jazayeri, Alex Gordon

Professor Johnson

CMSI 4071 Senior Project I

23 October, 2024

Pilone and Miles Questions

1. We found that the two major concerns of any software project are how much it will cost throughout and how long it will take to develop. In our opinion, the concern that is most important is the budget of the project. If there is no budget on a project, there will not be provided resources to develop and expand beyond a small starting line. On the other hand, if the budget is too large, then the project's scope can be unchecked and mismanagement of funds can be possible. Balance between these sides of budget checks can allow for a flourishing project and a better overall development experience. Complete functionality fits with the idea of balancing these two concerns to allow the project to run smoothly on budget and on time.

2. In the Agile method for software development, we found that the five main phases that occur in each and every iteration are: brainstorm, design, develop, provide quality assurance, and finally deployment of the project. We do not feel that any of the phases can only be done at the start of the project in comparison to the end since no project is perfect. There will inevitably be criticism and feedback on any design given when deployed and, therefore, going back to the project's roots while brainstorming solutions, redesigning, and redeveloping ideas is a necessary step. This would then lead into more

quality assurance and redeployment that will improve the project in future iterations. This comes at a cost though, with each iteration taking time and adding to the total length and possible budget of the project. Each restructuring comes with a cost but, if done well, could allow for the project to be better in the long run.

3. We found that the main phases that occur in the Waterfall method for software development include phases for requirements analysis, product design, system design, coding, testing, and maintenance. These phases are different from the Agile method because the phases include more time for building a requirements analysis document instead of brainstorming, splits designing the project into both product and system design instead of just design, replaces quality assurance with testing, and adds maintenance after deployment instead of restarting the development cycle entirely. The phases in the Waterfall method left out of the Agile method include the brainstorming phase, the quality assurance phase, and the specific step of the deployment phase. For waterfall, we think these stages are needed in waterfall because it would be removing general design thinking along with product maintenance and quality control from an undeployed system. For an Agile development cycle, the stage of brainstorming could be necessary if the project is running in circles and a solution for a task can not be found. The project may need to go back to basics and rethink their solution to their original problem.

4. The concept of a user story can be defined as a requirements list for what the user should be able to do with a product broken down into specific tasks and goals for development. This could be included with a priority number, to highlight importance, a time estimate

for how long development will take for the task, and a given title for the user story. Blueskying can be described as coming up with ideas to solve requirements which can be as large and wild as necessary as long as it focuses on the core goal the software is trying to meet. For this concept, the skies the limit when it comes to idea thinking as long as it achieves the goal with our software development. Four things that users stories should do includes describing one thing that the software needs to do for the customers, be written using language the customer understands, be written by the customer, and be short: no more than three sentences. Three things that user stories shouldn't do include not being a long essay, not using technical terms that are unfamiliar to the customer, and not mentioning specific technologies used for development. The Waterfall method does not have user stories with its specific steps and ideas.

5. In our opinion, we disagree on the first statement "All assumptions are bad, no assumptions are good assumptions" because it does not cover all situations. Assumptions can be bad in certain capacities placing stress and unrealistic values on certain goals and tasks. On the other hand, assumptions can give a bar to reach in terms of what is possible. We can assume that we have the ability to complete a project but may run into a wall when working on it and are encouraged to grow and move past it. Assumptions can be used to set a bar to reach and allow for a person to reach goals and have status in accordance to what they can do and what is possible with their skills. For the second statement "A "big" user story estimate is a "bad" user story estimate," we are mixed in terms of our opinions. On the one hand, having too much information in a user story can set unrealistic expectations for a product or prevent creative ideas from working as a

solution to problems throughout development. On the other hand, a large user story can also give structure and outlined goals with a project and allow for a more clear development path ensuring that the customer gets everything they want out of a developed product. These two sides of a user story must be balanced to allow for a good product for both developers and the customer at the end of the development cycle.

6. * 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 confident: **Estimate**

   * Everyone's involved when it comes to me: **Blueskying**

   We agree with every answer given in the book even when the book gave multiple answers for one question we did not think of. Observation makes sense for both statements 3 and 4 as a learning mechanism to understand everything in a question and get more out of a customer.

7. The concept of a better than best case estimate can be described as a prediction that surpasses the most optimistic scenario you usually consider in planning. It implies outcomes that are even more favorable than what you'd expect under ideal conditions. While it can be uplifting and point to potential surprises, it also risks setting overly

ambitious goals that might not be realistic therefore it is very important to manage expectations.

8. In our opinion, we think that the best time to tell a customer that you will NOT be able to meet the delivery schedule would be either early in the process or at the midpoint of the process if an extension is required. These two times are best because, if done early in the project, deadlines can still be rearranged and not a lot of effort and budget has gone behind an infeasible project. If this does not work, you would want to do it as soon as possible if you find it's impossible to meet the deadline and before all the budget is spent. If you do it at a midway point, there is still time to get possible extensions or back out of a project before everything is lost and the project is ruled a complete loss. This would obviously be a difficult discussion because you would have to go back on your original word and agreement to the customer with the deadline and there may have to be budgetary constraints that would have to be considered if the project was to be continued or shut down. To make it less difficult, you could come in with good stats and progress reports of your project to show how far it's gotten and come in with information on why the time constraints will not work for the task at hand. Being prepared to have the discussion instead of walking in empty handed will save you in the long run.

9. We as a group decided that branching in your software configuration is a good thing overall since it provides stability and recovery to code being provided. When there is a new version of the software you are working on and it must be maintained outside of the main development cycle, branching can allow for updates to be simple by allowing for

easy adding onto the already established projects. Branching also allows for stability of code so that, if a radical change is pushed and breaks the system, the team developing the project can just reverse the branch and thereby reset to the earlier version. Branching can also allow for each developer on a project to have their own separate branch that can be checked and monitored by the other developers before being pushed. This allows for easy progress reports for each developer that can either be accepted or denied depending on the overall group opinion. One scenario to support this opinion could be if someone pushes broken code to a repository that doesn't compile and breaks the overall system which can be reversed or caught early through the use of branching and developer approval.

10. Throughout our experiences of development, none of us have used build tools. From what we have heard and researched, some benefits are that it can save tons of time and minimize human error, however some drawbacks are that it may be complex and resultantly require a learning curve.